TEXT CUBE: CONSTRUCTION, SUMMARIZATION AND MINING

BY

FANGBO TAO

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair
Professor Jian Peng
Dr. Haixun Wang
Professor ChengXiang Zhai

Abstract

A large portion of real world data is either text or structured (*e.g.*, relational) data. Such data objects are often linked together (*e.g.*, structured product information linking with their descriptions and customer reviews.). To systematically analyze large numbers of such textual documents, it is often desirable to manage the text data with the associated structrued data in a multi-dimensional space (hence *text cube*).

This thesis studies the multi-dimensional representation of large textual data. Since Jim Gray introduced the concept of "data cube", data cube, associated with online analytical processing (OLAP), has become a driving engine in data warehouse industry. By modeling a large textual corpus as a "cube", *i.e.*, multi-dimensional and hierarchical structure, we bridge the power of traditional OLAP and Information Retrieval / Natural Language Processing techniques. In particular, this thesis focuses on two lines of work, one is to construct a multi-dimensional text cube from raw text data with limited user guidance; the other is to develop effective summarization and mining techniques tailored for multi-dimensional queries on text cubes.

In the first part of the thesis, the problem of *dimension-based structure creation* is studied. We propose an end-to-end framework for extracting multi-dimensional structure from a corpus, taking the input of a corpus of specific domain and limited seeds to generate a high-quality dimension values as output. We introduce the novel concept of Semantic Pattern Graph to leverage web signals to understand the underlying semantics of lexical patterns, improve pattern evaluation using mined semantics, and yield more accurate and complete structure. Experiments show the effectiveness of our approach.

In the second part, with all the dimensions discovered, we study the problem of *cell-based document allocation*. That is, linking the created dimensions with text data and construct a multi-dimensional text cube. To allocate documents into correct multi-dimensional subsets, *i.e.*, a cell. Traditional approaches, in this particular task, may require substantial labeling from user. Instead, we propose a model that requires no additional training data besides the given (label) name of each cube dimension as weak supervision. With such weak supervision, we develop a *dimension-aware joint embedding* framework that learns joint representations for terms, documents, and labels. In the joint embedding process, our method iteratively learns dimension-aware document representations by selectively focusing on discriminative keywords for different dimensions. Furthermore, it alleviates label sparsity by leveraging label representations to enrich

the labeled term set. Numerical experiments corroborate the effectiveness of our solution.

In the third part, we introduce the concept of *Context-Aware Semantic Online Analytical Processing* (*i.e.*, *CASeOLAP*) in text cubes, and use *top-k representative phrases* to represent the semantics of the document subset in a text cube cell. By ranking phrases with a newly proposed ranking measure according to three criteria: integrity, popularity and distinctiveness. We identify phrases that can successfully digest the main content of a subset of documents of interest and contrast with other neighboring subsets. Our experiments in a large news dataset demonstrate the effectiveness of the newly proposed ranking measure in finding representative phrases and the efficiency in both query processing time and storage cost. The approach is also applied to clinical biomarker analysis and protest news analysis with success.

In the last part, the system of *EventCube* is proposed to support end-to-end pipeline of text cube in an informative, interactive, and user-friendly manner. The system serves as a general platform for construction, search, summarization, OLAP (online analytical processing) and data mining on integrated text and structured data. The system is a growing testbed for various text cube based research and has been successfully applied to NASA for aviation safety report analysis and Army Research Lab for Counter-Terrorism Report analysis.

To summarize, this thesis provides important results of construction and consumption of multi-dimensional text cubes and shows its power in tackling real-world text analysis tasks.

To my parents and sister.

Acknowledgments

This thesis would not have been possible without the support of many people.

I am profoundly grateful to my advisor Dr. Jiawei Han, for his guidance and support on every aspect of my Ph.D. life. He inspired and encouraged me to find meaning in research, stay hungry for challenges, stand my ground and argue with grace. Of greater import to me is, his high character, courage, curiosity, grit and compassion, influenced me to be the best I can and live life with a positive purpose. I feel extremely lucky to be a member of the data mining group led by Dr. Han.

I greatly appreciate the guidance and support of my Ph.D. committee, Dr. ChengXiang Zhai, Dr. Jian Peng, and Dr. Haixun Wang, and am grateful to them for their invaluable comments on my dissertation.

Many thanks to my friends and research collaborators:

Dr. Yan Ke, Dr. Hao Ma, Dr. Bo Zhao and Dr. Ariel Fuxman from Microsoft Research, Dr. Lance Kaplan, Dr. Tim Harranty, Dr. Clare Voss, Dr. Taylor Cassidy from Army Research Lab, Dr. Venky Ganti from Alation, inc, Dr. Heng Ji, Dr. Yizhou Sun, Chao Zhang, Honglei Zhuang, Dr. Meng Jiang, Huan Gui, Jialu Liu, Chi Wang, Marina Danilevsky, Yang Li, Xiang Ren, Xiao Yu, George Brova, Tobias Lei, Qi Wang, Keyang Zhang, Dr. Quan Yuan, Zeqiu Wu, Frank Xu, Meng Qu, Brandon Norick, Xide Lin, Quanquan Gu, Jingjing Wang, Ahmed El-Kishky, Stephen Macke, Xiao Cheng, Lidan Wang, and every DAIS group member, who made my academic life much more enjoyable by providing helpful discussions and in standing by me during hard times.

Finally, and above all, I would like to thank my family for their endless love, support and encouragement. Thanks for giving me courage and confidence to pursue my dream. In every moment of crisis, their unwavering love has guided me through until this moment. Special thanks to my girlfriend, Xuran Peng, for your tremendous support during this thesis. I dedicate my thesis to all those mentioned.

This thesis was supported in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of the U.S.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

We are living in the big data age. A large portion of real world big data is either text or structured data. Moreover, such data objects are often linked together (*e.g.*, product meta-information linking with customer reviews and flight information linking with its aviation safety report). Researchers in information retrieval and natural language processing have been dedicated to provide understanding and intelligence to text corpora. Applications such as *topic modeling*, *summarization* and more have been popular and effective. However, most text-based techniques concentrate on a single document or a set of document with no internal structure. With associated meta data and structure between document groups been neglected, the aggregative and comparative power of text analysis is largely weakened.

On the other hand, *data cubes* have been constructed popularly on structured data, especially numerical data, to facilitate online analytical processing (OLAP) of multi-dimensional databases. An OLAP data cube organizes data with categorical attributes (called *dimensions*) and summary statistics (called *measures*) from lower conceptual levels to higher ones. By providing users the ability to access data collections of any dimensional subsets (called *cells*), a data cube offers ease and flexibility for data navigation and analysis by different granularities and from different perspectives. Nevertheless, traditional data cube is not able to manage large text data and it only focuses on numeric measures, such as *text summarization*, cannot be easily supported.

With ever more massive datasets accumulating in text repositories (*e.g.*, news articles, business reports, customer reviews, *etc.*), it is highly desirable to conduct multi-dimensional analysis on text data. Some recent studies introduce text cube or topic cube [37, 68], where the dimensions correspond to multiple meta attributes (*e.g.*, category, date/time, location, author, *etc.*) associated with the documents, whereas the measures correspond to some numerical measures (*e.g.*, count, probability, *etc.*) associated with a set of keywords. An example of such text cube is as follows.

Example 1.1 ***Text Cube.*** *Suppose a multi-dimensional text cube is constructed from a* New York Times *news article repository with three meta attributes:* Location*,* Topic*, and* Time*. Each attribute form a*

hierarchical dimension where the possible values are in the hierarchy. For example value Illinois *in* Location *is the child of value* US. *Each document is in the form of have one specific value in each dimension along with its text content. An analyst may pose two multi-dimensional queries: (q_1): $\langle$*China, Economy*$\rangle$ and (q_2): $\langle$*US, Gun Control*$\rangle$, corresponding to two dimensions* Location, *and* Topic. *Measures such as* frequent word counting *can be returned as a measure.*

To model a large textual corpus as a "cube", we focus on two tasks. The first task is the construction of text cube given a raw text corpus. Such formal and structural representation of information has the advantage of being easy to manage and reason with. From raw but rich textual signals, we aim to discover the hidden cube structure. To avoid costly human labeling, we devise approaches that only take limited user guidance and construct cube automatically. The second task is the development of advanced text-specific measures. Instead of simple counting measures as in traditional OLAP, we propose measures that consider the deep semantic of documents and can better serve as comparative and aggregative information for document understanding.

1.2 Text Cube Construction

The example above assumes the multi-dimensional structure is provided beforehand, whereas real scenarios rarely have cube structure fully constructed by human. Therefore, the first theme of this thesis is cube construction. Specifically, the ultimate goal of cube construction is 1) finding all the dimensions, and 2) assigning every document into the right cell (*i.e.*, multi-dimensional subset). Therefore, a two-step framework is proposed to construct multi-dimensional text cube from raw document corpora.

- **Step 1: Dimension-based Structure Creation** Given partial dimension values for each dimension, discover the entire set of dimension values. For example, given *Apple* and *Samsung* as seeds in dimension *Brand*, discover all significant electronic brands from customer review text.
- **Step 2: Cell-based Document Allocation** Given dimensions and their value sets, assign every document into the multi-dimensional space framed by dimensions. For example, given location dimension (*e.g.*, China, USA, UK, *etc.*) and topic dimension (*e.g.*, Economy, Sports, Politics, *etc.*), each news article is assigned to a location-topic pair (*i.e.*, a *cell*).

Since the target corpora that our framework handles are mostly from specific domains, *e.g.*, news, safety report and government documents. In contrast to "open domain", which is the alias for Web, we characterize our target data as "closed-domain" corpus in the remaining thesis.

1.2.1 Dimension-based Structure Creation

The *Dimension-based Structure Creation* task is very related to the Web Information Extraction [21, 9, 29, 52] problem, which aims to extract interesting entities (and potentially the relations among them) from Web corpus. Unfortunately, the techniques proposed for Web entity extraction can hardly be applied to closed-domain scenarios. This is because closed-domain corpora exhibit very different characteristics compared with the Web corpus. It mainly suffer from semantic drifting of entity names and sparsity of lexical patterns that identifies entities.

To address these two problems, we developed a semi-supervised pattern extraction method and leverage Web data to understand the pattern semantics. Semi-supervised pattern extraction can help extract more lexical patterns from closed-domain text, while Web text can be used to build semantic relations between extracted patterns, which can be further utilized to induce sparse yet effective patterns. The built *Semantic Graph* helps to identify bad general patterns and good sparse patterns mentioned above, thus fixing the sparsity issue.

We tackle the problem via bootstrapping. We take a few seeds plus an unlabeled corpus as input and produce a high-quality entity set as output. With insufficient knowledge of lexical patterns in closed-domain corpus, we leverage public signals (i.e., Web text) to understand the underlying semantic relations among sparse lexical patterns. For this purpose, we propose a novel concept called Semantic Pattern Graph (SPG), which is a hierarchical graph structure describing the relations between lexical patterns. To the best of our knowledge, this is the first work that models semantic relations between lexical patterns for entity extraction. Experimental results show that the proposed framework achieves the best precision and coverage balance of dimension values. The detailed approach is discussed in Chapter 3.

1.2.2 Cell-based Document Allocation

After *Structure Creation* is finished, the dimensions of interests and their values are discovered. However, the structure data and text data are still disconnected. To enable the power of multi-dimensional text analysis, *Cell-based Document Allocation* is required to allocate documents into the multi-dimensional cell space. Simply put, the goal of *Cell-based Document Allocation* is to find the most proper dimension value for each document in each dimension. This task is closely related to text categorization [1], yet there are notable differences between the two problems. Text categorization is typically formulated as a classification problem: assuming a corpus of training documents is given, the process is to extract different features for those documents and learn a classifier that can infer the ground-truth labels for any documents. Cell-based document allocation, on the other hand, does not have labeled training documents in most scenarios. More

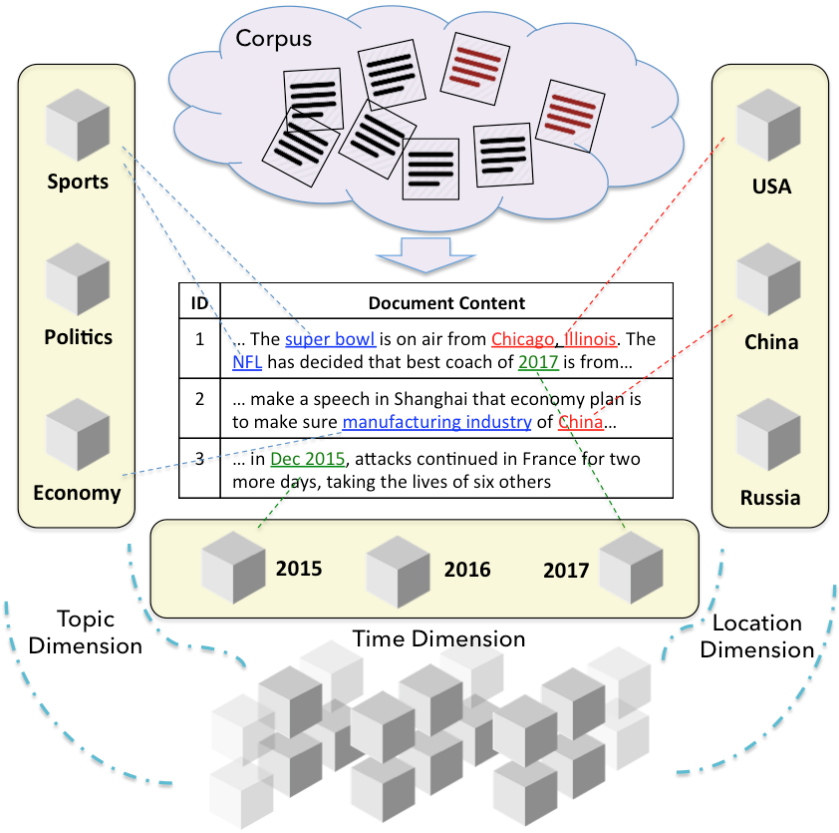


Figure 1.1: Illustration of cell-based document allocation on news articles with three dimensions: topic, location and time.

importantly, cell-based document allocation is a multi-dimensional categorization problem in nature. For example, Figure 1.1 shows an illustrating task of news cube construction. As shown, to assign each news article into a cube cell (*i.e.*, a multi-dimensional label like $\langle$*Sports, 2017, USA*$\rangle$), we need to consider different dimensions simultaneously: 1) the topic dimension: what is the event about? 2) the time dimension: when does the event happen? and 3) the location dimension: in which country does the event happen? To neatly organize such a news corpus, it is necessary to perform cell allocation for each article and selects an appropriate label from every cube dimension.

We propose a *dimension-aware joint embedding* method for effective cell-based document allocation. Our method does not require any costly human labeling process. Instead, it simply uses the label names in different cube dimensions as a small set of labeled seed terms. Regarding these seed terms as weak supervision, we develop a dimension-aware joint embedding framework that learns different embedding spaces for different cube dimensions. As a result, each document can have multiple representations, each tailored for one cube dimension to achieve high discriminative power. The detailed model is discussed in

Chapter 4.

1.3 Text Cube Summarization and Mining

The second theme of this thesis is the multi-dimensional summarization and mining techniques on text cubes. In contrast to traditional data cube technology which focus on numerical data, text cube carries more comprehensive and subtle information. Therefore, besides numerical measures like *sum* and *average*, various in-depth text-based measures can be developed. Moreover, compared to standard text analysis methods, text cube naturally have rich structure. By exploring both text space and structural cell space, one can be apply text measures in a comparative and structural way for better understanding. In this thesis, we focus on two types of multi-dimensional queries.

- **Point Query:** A point query in text cube is a multi-dimensional query consists of one or more predicates on dimension values. The query specify the target document subset that user want to perform analysis on. The measure associated with a point query is evaluated solely based on the target document subset.
- **Comparative Query:** A comparative query is a point query with a set of comparative cells. The measure associated with a comparative query is evaluated based on the comparison of the target document subset and its comparative groups.

Since a text cube may handle very large text corpus, *e.g.*, millions of documents, efficiency is another critical performance measure for real-time analysis. En route to improving effectiveness of proposed in-depth measures, several optimization techniques are developed to ensure the search, summarization and mining on text cube can be conducted interactively and in real-time.

1.3.1 Comparative Analysis via Representative Phrase Mining

We introduce a new concept called *Context-Aware Semantic Online Analytical Processing* (*i.e.*, *CASe-OLAP*) in text cubes, that is, online dynamic presentation of the major semantics of the cube cells in multi-dimensional context. Clearly, the semantics of a cube cell presented should characterize the set of documents in the cell and be distinguished from that of other cells (*i.e.*, context) in the cube, which leads to our proposal of using *top-k representative phrases* to represent the semantics of a text cube cell. Let us examine an example.

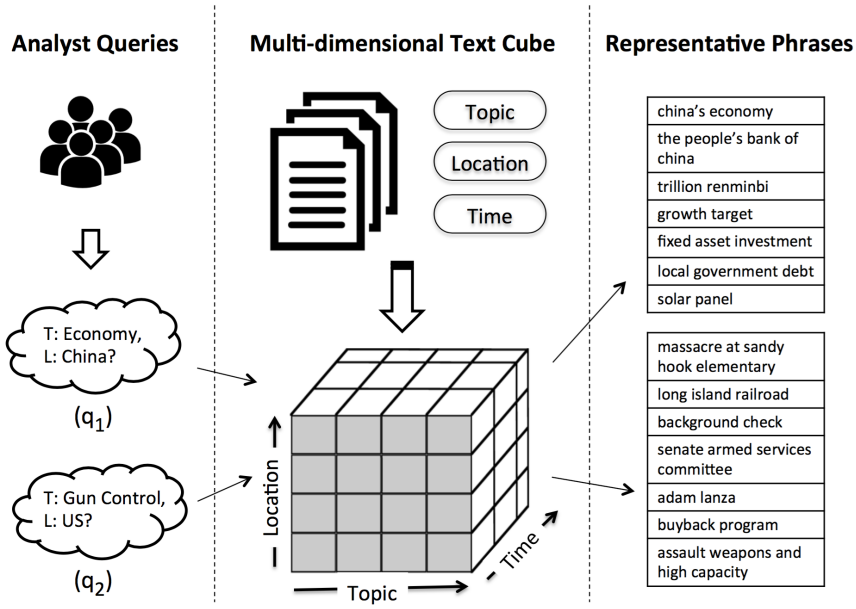


Figure 1.2: Illustration of representative phrase queries in multi-dimensional text cube

Example 1.2 *Suppose a multi-dimensional text cube is constructed from a* New York Times *news article repository with three meta attributes:* Location, Topic, *and* Time. *An analyst may pose two multi-dimensional queries: (q_1):* ⟨China, Economy⟩ *and (q_2):* ⟨US, Gun Control⟩, *corresponding to two dimensions* Location, *and* Topic. *What kind of cell summary output do we like to see? Frequent unigrams such as* debt *or* senate *are not very revealing, whereas multi-word phrases, such as* local government debt *and* senate armed service committee, *capture the key information and present better semantics. Moreover, the key-phrases should be representative of this cell and distinct from other cells, that is, top-k representative phrases, as shown in Figure 1.2.*

The above discussion leads to three criteria for ranking representative phrases in a selected cube cell: (i) *integrity*: a phrase that provides integral semantic unit should be preferable over a unigram, (ii) *popularity*: popular in the current cell (*i.e.*, current subset of documents), and (iii) *distinctiveness*: distinct from other cells.

In this work, we systematically build our solution based on these insights. First, SegPhrase is employed to generate candidate phrases for the entire corpus. Second, a ranking measure is designed to respect all the three criteria. Specifically, the measure incorporates phrase distribution information from sibling cells (*e.g.*, ⟨Japan, Economy⟩) as context. The CASeOLAP framework is also applied to multiple real world scenarios and promotes productivities for multiple groups. Two cases will be discussed in Chapter 5. One is the collaboration with UCLA on *Clinical Biomarker Analysis* , the other is the collaboration with RPI on

Protest News Analysis.

1.3.2 System Design: Multi-dimensional Search and Mining

In the last piece of this thesis, we develop an efficient and easy-to-use system, *EventCube*, to implement the end-to-end pipeline of multi-dimensional text cube. Typically, structured/relational data has been handled by relational database systems, and such systems also provide some text indexing and search capabilities to assist text data stored in such (extended) relational database systems. However, such kind of systems often suffer from the following limitations.

1. It does not support the generation of dimensions from raw text data. Users are required to prepare all metadata beforehand.

2. It can hardly support systematic search and analysis of large collections of free text in multi-dimensional way, although text data is ubiquitous in real-world;

3. It usually does not support data cube technologies on text data and multidimensional text mining although it is obvious that text mining and data cube technologies can mutually enhance each other; and

4. There is a lack of a general platform that can support integrated multi-dimensional analysis of structured and text data, on top of which many powerful analysis methods and tools can be developed, experimented and refined, such as viewing such data sets as interconnected information networks and further applying information network analysis technology.

EventCube is a project that provides such a general platform that can easily import any collection of free text, such as news data, aviation reports or academic papers, extract entities and topics, construct the multi-dimensional text cube and support powerful search and mining functions. For structured data, users can load pre-extracted dimensions to construct the text cube. For text-intensive data with minimally predefined structured information (e.g., news data), built-in structure creation and document allocation tools are used to extract entities serving as dimensions and assign documents into the cube. This framework provides a tremendous opportunity to conduct multi-dimensional analysis on text and structured data in powerful and flexible ways. This great potential motivates our research and development of the EventCube system for multidimensional search and analysis of interconnected structured and text data or on text data via cube construction using entity extraction and dimension building tools.

1.4 Text Cube Basics

Similar to traditional multi-dimensional data cubes, a *text cube* [37] is a data model but over text collection $\mathcal{DOC}$ that has metadata for documents. The metadata can be either extrinsic attributes of the documents, such as classification taxonomy, or intrinsic information extracted from the documents, such as named entities mentioned in them. In this work, we focus on single-valued categorical metadata, and leave other types of metadata to future work. We assume there are n categorical attributes (*i.e.*, *dimensions*) associated with each document in $\mathcal{DOC}$. For example, a news article in *NYT* corpus is represented as (*Jan 2012*, *China*, *Economy*, *'After a sharp economic slowdown through much of last year...'*). It denotes that the *'Time'* of the article is *Jan 2012*, *'Location'* is *China* and *'Topic'* is *Economy*. We formally define a dimension as follows:

Definition 1.1 (Dimension) *A dimension in a text cube is defined as* $\mathcal{L} = \{l_1, l_2, \ldots, l_{|\mathcal{L}|}\}$*, where* $l_i \in \mathcal{L}$ *is a label (*i.e.*, dimension value) in this dimension. The labels are organized in either flat way, or hierarchical way.*

For i-th dimension, the dimension $\mathcal{L}_i$ is a tree where the root is denoted as '$*$'. Each non-root node is a value in that dimension. The parent node of a dimension value l_i is denoted as $par(l_i)$, and the set of direct descendants of l_i is denoted as $des(l_i)$. For example, Figure 1.4 illustrates a partial dimension hierarchy about 'topics' in *NYT* corpus. It is a tree of height 4, with a root node '$*$'. $par(Gun\ Control) = Domestic\ Issues$ and $des(`*') = \{Economy, Sports, Politics\}$.

Formally, we have the following definition.

Definition 1.2 (Text Cube) *A text cube is defined as* $\mathcal{TC} = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n, \mathcal{DOC})$*, where* $\mathcal{L}_i$ *is the* i*-th dimension. Each document* $d \in \mathcal{DOC}$ *is represented as* $(l_{t_1}, \ldots, l_{t_n})$*, where* l_{t_i} *is the label of* d *in dimension* $\mathcal{L}_i$*. A cell* c *in the cube is represented as* $(l_1, \ldots, l_n, \mathcal{D}_c)$*, where* $l_i \in \mathcal{L}_i$*, and* $\mathcal{D}_c \subseteq \mathcal{DOC}$ *is the subset of documents contained in cell* c*. For notation simplicity, we use* $\langle l_{t_1}, \ldots, l_{t_n} \rangle$ *to refer to a cell with non-*$*$ *dimension values* $\{l_{t_1}, \ldots, l_{t_n}\}$*.*

An illustrating example of text cube on *New York Times* news articles is shown as follows.

Example 1.3 *Fig. 1.3 illustrates a mini example of news article text cube, with 3 dimensions (Time, Location and Topic) and 9 documents* d_1*–*d_9*. The Time dimension is derived from extrinsic attribute but Location and Topic are constructed using our proposed cube construction framework. . We list 7 non-empty cells, where the top four are leaf cells without '*$*$*' dimensions,* e.g.*, (Jan 2012, China, Economy,* $\{d_1, d_2\}$*). The root cell (entire corpus) is represented as (*$*$*,* $*$*,* $*$*,* $\{d_1$*–*$d_9\}$*).*

Dimensions			Text Data
Time	**Location**	**Topic**	$\mathcal{DOC}$
Jan 2012	China	Economy	$\{d_1, d_2\}$
Aug 2012	China	Economy	$\{d_3, d_4, d_5\}$
Aug 2012	US	Gun Control	$\{d_6, d_7\}$
Nov 2012	US	Economy	$\{d_8, d_9\}$
$*$	China	Economy	$\{d_1, \ldots, d_5\}$
Aug 2012	$*$	$*$	$\{d_3, \ldots, d_7\}$
$*$	$*$	$*$	$\{d_1, \ldots, d_9\}$

Figure 1.3: Mini Example of *NYT* Corpus

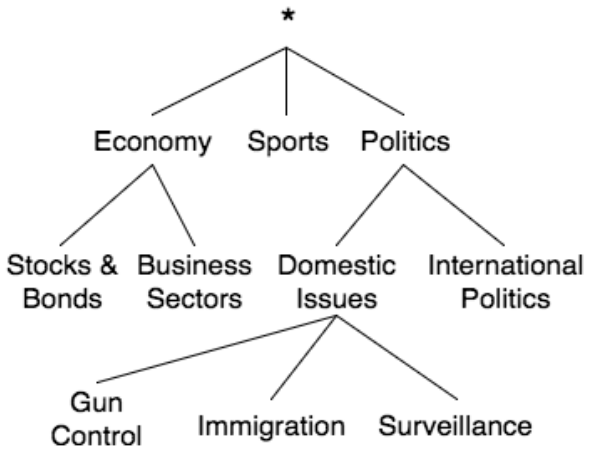


Figure 1.4: Hierarchy of *Topic*

Text cube provides a framework for organizing text documents using meta-information. In particular, the cell space defined above embeds the inter-connection between different subsets of text. To capture those semantically close cells, we define *context* of a cell c as a composition of three parts.

Definition 1.3 (Cell Context) *The context of cell* $c = \langle l_{t_1}, \ldots, l_{t_n} \rangle$ *is defined as* $\mathbb{P}(c) \bigcup \mathbb{S}(c) \bigcup \mathbb{C}(c)$, *where:*

- *Parent set is defined as* $\mathbb{P}(c) = \{\langle l_{t_1}, \ldots, par(l_i), \ldots, l_{t_n} \rangle | i \in t_1, \ldots, t_n\}$. *Each parent cell is found by changing exactly one non-$*$ dimension value in cell c into its parent value;*
- *Children set is defined as* $\mathbb{C}(c) = \{c' | c \in \mathbb{P}(c')\}$. *Each child cell is found by either changing one $*$ value into non-$*$ or by replacing it by one of the child values; and*
- *Sibling set is defined as* $\mathbb{S}(c) = \{c' | \mathbb{P}(c) \bigcap \mathbb{P}(c') \neq \emptyset\}$. *Each sibling cell must share one parent with cell c.*

Example 1.4 *Fig. 1.5 illustrates the partial context of cell* $c = \langle$*China, Economy*$\rangle$. *The parent set* $\mathbb{P}(c)$ *contains* $\langle$*China*$\rangle$ *and* $\langle$*Economy*$\rangle$, *sibling set* $\mathbb{S}(c)$ *has* $\langle$*China, Politics*$\rangle$ *and* $\langle$*US, Economy*$\rangle$ *and children* $\mathbb{C}(c)$

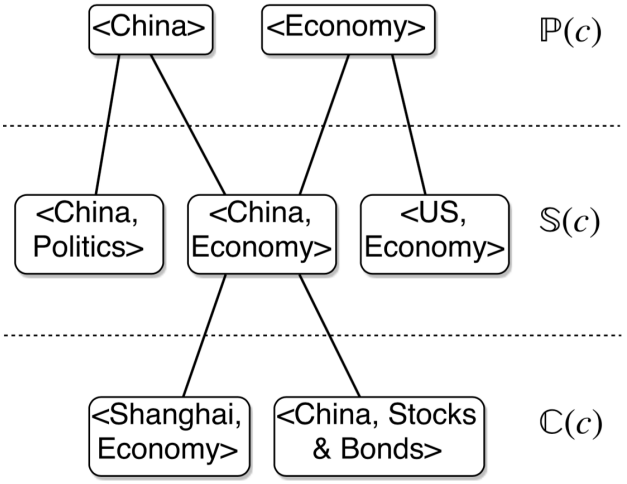


Figure 1.5: *Context* of cell ⟨China, Economy⟩

contains ⟨*Shanghai, Economy*⟩ *and* ⟨*China, Stocks & Bonds*⟩.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, the literature review is presented. In Chapter 3, a dimension-based structure creation solution is proposed to bring structure into unstructured corpus. In Chapter 4, a dimension-aware joint embedding model is presented for multi-dimensional cell-based document allocation. In Chapter 5, context-aware semantic OLAP is investigated, where the representative phrases are mined given a multi-dimensional query in text cube. In Chapter 6, EventCube system is detailed. Chapter 7 lays out the future research. Chapter 8 concludes the thesis.

Chapter 2

Literature Review

2.1 Text Cube

Text Cube proposed in [37] introduces the concepts of dimension and cube to the analysis of large sets of document and studies aggregation of textual measures in multi-dimensional space. Since then, text cube mining has drawn much attention [68, 60]. For text cube analysis, several important pieces of related work [54, 4, 38] have been introduced in Section 1. Here we discuss other previous work related to CASeOLAP. [34, 47] proposed OLAP-style measures on term level using only local frequency, which cannot serve as effective semantic representations. [70, 18] focused on interactive exploration framework in text cubes given keyword queries, without considering the semantics in raw text. Several multi-dimensional analytical platforms [42, 59] are also constructed to support end-to-end textual analytics. However, the supported measures are numerical term-level ones. Another related topic is Faceted Search [33, 62, 5, 16], which dynamically aggregates information for an ad-hoc set of documents. the aggregation is usually conducted on meta data (called *facets*), not document content. Moreover, these studies focus on the text analytics tasks assuming that the cube structure is constructed by data provider. The text cube construction task remains largely unsolved.

2.2 Structure Extraction from Text

Recently, several open information extraction systems have been created. Most of them focus on entity and relation extraction on web-scale data. Knowitall [21, 22, 19] combined pattern-based and list-based extraction to achieve recall improvement. They used a set of generic, domain independent extraction patterns (mostly Hearst patterns [32]) to extend a set of seed concepts. NELL [9] is another open information extractor for harvesting entities and relations from the web. It used Coupled Pattern Learner [10], which extracts lexical patterns with part-of-speech (POS) restrictions from positively labeled data, to identify new entities. A restrict filtering constraint is applied to guarantee only high-precision/low-recall patterns are promoted. SPIED [29] used a similar way of pattern generation but scored patterns using both labeled and unlabeled entities. These three systems were designed to work with redundant entity mentions (e.g., web data) and

performed poorly on sparse closed-domain settings. In our tests of these pattern generators, we observe that high-precision/low-recall patterns they used can hardly cover 50% of our target entity set. We provide labeled data for SPIED and both precision and recall are quite unsatisfactory. Other open information extraction systems like ReVerb [24, 23] and OLLIE [52] used verbal patterns or some extraction templates, which may fail to work well in sparse environment. Moreover, Poon and Domingos [46] showed that open information extraction systems extracted low accuracy relational triples on a small corpus. Hence, two things are required in *Dimension-based Structure Creation*: 1). More low-precision patterns need to be leveraged to cover more entities. 2). Open signals from the web need to be incorporated to improve accuracy of those sparse patterns.

Our *dimension-based structure creation* employs a semi-supervised bootstrap learning method, which begins with a small labeled set of target entities, trains a learning/ranking model, and uses that model to label more data and so on. Yarowsky [66] applied bootstrapped learning on word sense disambiguation. Later Riloff [48] used a set of seed entities to learn rules for entity extraction from unlabeled data and extended it to multi-class learning in [61]. Similar methods are also used in many set-expansion works. SEAL [63, 64] is a web-based set expansion system that uses wrappers (i.e., page-specific extraction rules) to extract more entities. It takes advantage of both page structure and text from webpages. For many cases where only text data is available, several IE systems [29, 2, 8, 49] also applied bootstrap method, specify a small set of domain-specific seed instances as input, then alternately learn patterns from seeds, and extend seeds from patterns.

A distinctive feature of our *structure creation* solution is its use of Semantic Pattern Graph, which is derived from web-scale data, to re-score patterns and ease the sparsity issue. The pioneering work for pattern generation and scoring by Hearst [32] manually evaluated generated patterns to extract hypernym-hyponym pairs. Previous systems [25, 14, 56] used fully labeled corpus to score rules. Later, Carlson et al. [10] assessed patterns by precision and only promote patterns with high precision. We implemented their pattern assessment method for comparison and shown that his strategy works poorly on small corpus due to the coverage of high-precision patterns are very small. Gupta and Manning [30, 29] predicted labels of unlabeled entities to score patterns using features like distributional similarity and edit distances. None of the above works well enough in a closed-domain setting due to the sparse and biased signals of patterns. Our system outperforms them by utilizing outside signals from the web to adjust the biased evaluation of patterns, which is computed merely using mentions in closed-domain corpus.

2.3 Bridging Documents and Structure

Cell-based Document Allocation is closely related to text categorization. Most machine learning categorization methods take a supervised approach based on *e.g.*SVM, decision tree [1, 53], or neural networks [65]. A large collection of labeled training documents are often required for learning reliable classifiers. Our model, on the other hand, does not require labeled documents, but simply takes label names as weak supervision. Some studies have also explored the idea of unsupervised or weakly-supervised methods that merely use label names. [35] uses heuristic rules to generate training data, but the labels need extra feature engineering efforts to ensure the quality. OHLDA [31, 13] applies topic model with given labels to generate classifiers, 10 documents from *wikipedia* are extracted to represent the labels. Similarly, dataless methods [11, 55] use distant supervision to leverage *Wikipedia* to obtain a semantic vector for each label. The limitation of OHLDA and dataless models is their dependency on external knowledge bases. Therefore, these models cannot solve the categorization problem if the input corpus is closed-domain or has small coverages by external knowledge bases. Another limitation of these methods is their fixed representation of documents. The obtained document representations are dimension-agnostic and thus may not work well for all the cube dimensions.

2.4 Phrase Mining

The most related work to CASeOLAP is Multidimensional Content eXploration (MCX) [54], which studied phrase ranking for an arbitrary subset of documents. The system mines frequent word sequences using a frequency cutoff, and then ranks phrases based on phrase frequency ratio in the subset and in the whole collection. This system will find phrases that are popular, but not necessarily integral and distinctive. [38] studied quality phrase mining from a large corpus and proposed a SegPhrase approach. Phrases are mined globally using the whole corpus statistics, and text segmentation is used to respect integrity. The ranking of phrases is query-independent. If we simply combine MCX and SegPhrase, we can address integrity and popularity, but not distinctiveness.

Quality phrase mining is also extensively studied by NLP community [3, 69] and Data Mining community [15, 20]. They either utilize sophisticated NLP features or use various statistical measures to estimate phrase quality. Those phrase mining methods serve as the candidate generation step for our framework.

2.5 Representation Learning

Both text cube construction and consumption adopt the newly developed representation learning techniques. Here we discuss the previous representation learning methods and the novelty we bring to tackle representation learning in multi-dimensional setting. Since the success of word embedding approaches [43, 44], learning distributed representations as feature vectors [6] has been explored for different data types and applied to various tasks. One line of work closely related to our method is network embedding. Methods have been proposed for both homogeneous [58, 45] and heterogeneous graphs [57, 28] to preserve node correlations. While our *cell-based document allocation* method also relies on graph embedding, there are notable differences between our method and existing graph embedding methods. For existing graph embedding methods, their central theme is to find a static representation for graph nodes and thus fixed document embeddings. Such fixed document embeddings are dimension-agnostic and may not be optimal for many dimensions in the cube construction process. By contrast, our method considers the characteristics of different dimensions and obtains dimension-aware document embeddings to achieve high accuracies for all the dimensions.

Chapter 3

Dimension-based Structure Creation via Semantic Pattern Graph

3.1 Overview

The *Dimension-based Structure Creation* task is very related to the Web Information Extraction [21, 9, 29, 52] problem, which aims to extract interesting entities (and potentially the relations among them) from Web corpus. Unfortunately, the techniques proposed for Web entity extraction can hardly be applied to closed-domain. This is because closed-domain corpora exhibit very different characteristics compared with the Web corpus, which leads to the following unique challenges for extracting meaningful entities in closed-domain corpora.

1. **Data Sparsity:** Closed-domain entities exhibit much less redundancy than public entities in the Web. Therefore, the extracting techniques based on frequencies or statistics can hardly be utilized.

2. **Semantic Drift:** Many entities in closed-domains have special meanings different from the popular meanings in external world. For example, within Microsoft, "Blue" could mean "Microsoft Blue" instead of the color. Even for the same entities, the internal usage could be very different from the external usage. For example, for the entity "Windows", external usage mainly focuses how to install and use it, while internal usage cares more about implementation and design. Therefore, it is inappropriate to directly explore distributional semantics [36] for iteratively extracting entities.

3. **Low Public Coverage:** Most internal entities are not covered by public knowledge bases. Some are even not covered by the Web. Therefore, it is very hard to leverage public resources (e.g., Wikipedia, Google) to provide complementary information. For instance, the entity "Cloud ML" is an internal project that can only be found within Microsoft.

Data sparsity suggests to consider public data to alleviate deficient redundancy. However, due to Challenges 2 & 3, incorporating public signals on entities directly introduces much noise via Semantic Drift and also, not feasible for internal entities because of Low Public Coverage. Thus, instead of harvesting entity signals from public data, pattern-level approach is considered to tackle these challenges.

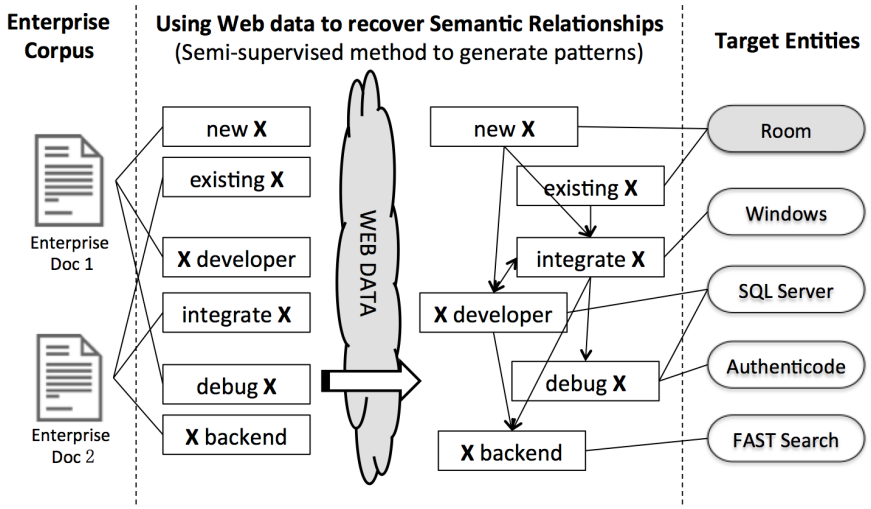


Figure 3.1: A toy example showing how we integrate closed-domain corpus with web data to construct semantic pattern graph and improve dimension quality.

Most previous studies [21, 9, 29, 52] utilize general high precision/low recall patterns (e.g. Hearst Patterns) to extract entities. Namely, they extract lexical patterns, optionally with some fixed form (Hearst Patterns), from the contexts of seed entities, evaluate the patterns' precision, and promote only a few top ones for further entity extraction. While this is a good fit for the redundant Web corpus, it causes severe coverage issues in structure creation task since the target corpus is very normally sparse. Based on our experiment on Microsoft corpus, under-utilizing these accurate patterns leads to less than 50% of meaningful values being successfully discovered. That is to say, most entities in closed-domain documents have contextual patterns with deficient mentions to be justified. But these patterns have highly relevant semantics to target entities, which we define as **Sparse Patterns**. Therefore, to better cover the remaining entities, some additional patterns (likely sparse) need to be incorporated.

To address these two problems, we developed a semi-supervised pattern extraction method and leverage Web data to understand the pattern semantics. As illustrated in Figure 3.1, semi-supervised pattern extraction can help extract more lexical patterns from closed-domain text, while Web text can be used to build semantic relations between extracted patterns, which can be further utilized to induce sparse yet effective patterns. The built *Semantic Graph* helps to identify bad general patterns and good sparse patterns mentioned above, thus fixing the sparsity issue.

We tackle the dimension-based structure creation problem via bootstrapping. We take a few seeds plus an unlabeled corpus as input and produce a high-quality entity set as output. With insufficient knowledge of lexical patterns in closed-domain corpus, we leverage public signals (i.e., Web text) to understand the underlying semantic relations among sparse lexical patterns. For this purpose, we propose a novel concept called

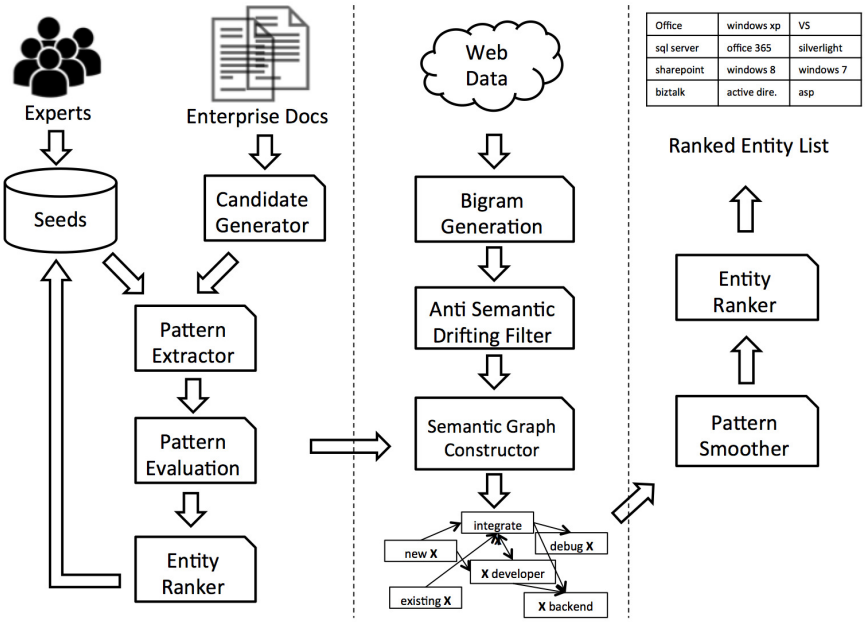


Figure 3.2: System Infrastructure

Semantic Pattern Graph (SPG), which is a hierarchical graph structure describing the relations between lexical patterns. To the best of our knowledge, this is the first work that models semantic relations between lexical patterns for entity extraction. Experimental results show that the proposed framework achieves the best precision and coverage balance of dimension values.

3.2 Preliminaries

We apply a set expansion framework to discover dimension values, dimension by dimension. The reason is that letting domain experts provide a seed set for each dimension is relatively cheap. Moreover, in some closed-domains (e.g., Microsoft), a partial taxonomy has been built by their employees, which can serve as the seed set directly. For ease of exposition, we present the approach below for extracting entities for one dimension $\mathcal{L}$. It can easily be generalized to multiple classes. The bootstrapping process involves following steps (also shown in Figure 3.2).

1. Generating candidate pool and labeling data: By scanning all the documents in the closed-domain corpus, a candidate pool containing all the possible closed-domain entities is generated. Positive entity seeds from dimension $\mathcal{L}$ is also provided by closed-domain insiders/experts. Negative seeds are automatically generated from the candidate pool using heuristics, i.e., capitalized ratio;
2. Pattern Extractor: Contextual patterns are created using context text around all the candidate entities

in both seed set and candidate pool.

3. Evaluating patterns using Multinomial Naive Bayes model.
4. Ranking entities and adding top confident ones into seeds. A semi-supervised framework is used to extract more low-precision/sparse patterns. Each iteration adds the top positive/negative entities into the seed set and re-scores patterns, after certain amount of iterations.
5. Semantic Pattern Graph Construction: Bi-grams are first extracted from web text for semantic recovery. An Anti Semantic Drifting Filter is then applied to avoid inconsistency of single pattern's semantic in different domains. And with the extended pattern list generated in closed-domain corpus, a Semantic Graph is built using bi-gram data and closed-domain patterns.
6. Smoothing pattern scores: A smoother is applied to re-evaluate each pattern based on their original score and graph structure.
7. Ranking entities: Smoothed patterns for the class are applied to the entity candidates. A Multinomial Naive Bayes classifier ranks the candidate entities and adds the top entities to $\mathcal{L}$'s dictionary.

3.3 Main Framework

3.3.1 Generation of Candidates & Patterns

Different from the previous Web-scale entity extractors, we generate all possible entity candidates as our first step. Many closed-domain entities and their patterns might be very sparse, such that using only seed patterns may never discover them. Finding these sparse entities first takes their sparse patterns into account in the process. Two restrictions are applied in the generation of candidate pool. First, the target term has a part-of-speech (POS) restriction, which is the POS tag sequence of the candidate phrase. Second, a noun phrase will be considered as a candidate only if the phrase appears at least once in the corpus as capitalized form.

In our framework, we automatically generate negative seeds from the candidate pool. For a candidate term t, sets C_t and U_t denote the capitalized and uncapitalized mention sets in the corpus. We require $\frac{|C_t|}{|C_t|+|U_t|} \leq threshold$ to consider t as a negative seed. We set $threshold = 0.1$ in our experiments.

We use lexico-syntactic surface word patterns to extract entities from candidate pool. They are created using contexts of words or their lemmatized form within a window of one before or after a entity candidate in the candidate pool. Here we collect as many patterns as possible, even include those that never appear with any positive/negative samples. These "invisible" patterns might be quite useful after understanding its

semantic relations in the smoothing phase. Therefore, we take them into account in every step. We generate flexible pattern by removing {"a", "an", "the"} when matching patterns to the text.

Two reasons are considered to pick one as the window size instead of two or more: 1) Bigger window size may introduce more noisy patterns that are irrelevant to the entity. 2) It is more interpretable to construct semantic pattern graph on unigram patterns.

To simplify the notation, we will use "+" or "−" to indicate the relative location of lexical patterns. "+" means the pattern appears after the entity and "−" means the pattern appears before the pattern. Thus, pattern **"existing X"** would be **"−existing"** and **"X developer"** would be **"+developer"** and so on.

3.3.2 Pattern Scoring with Semi-NB

A Semi-supervised Naive-Bayes (**Semi-NB**) model is applied here as the ranker and classifier for our task. We treat lexical patterns as features and treat every candidate in the candidate pool as testing data. User-provided positive seed set and automatically generated negative seed set are training data in our Semi-NB setting. For each target entity e in both training and testing data, a feature vector is generated, each feature score on feature f is calculated as:

$$S(e, f) = \log\left(count(e, f) + 1\right) \tag{3.1}$$

The $count(e, f)$ here means the total mention count of target entity e with feature (pattern) f. With the real-valued feature scores, we use a multinomial Naive Bayes to calculate pattern scores and entity score. Entity scores $T(e)$ are calculated as follows:

$$T(e) = \log\frac{P(+|e)}{P(-|e)} = \sum_f S(e, f)(\log P(f|+) - \log P(f|-)) + \log\frac{P(+)}{P(-)} \tag{3.2}$$

Where $P(f|+) = \frac{count(f,+)+1}{count(+)+|F|}, P(f|-) = \frac{count(f,-)+1}{count(-)+|F|}$. F is the set of all patterns. To evaluate each pattern, we also define normalized pattern score $R(f)$ as

$$R(f) = \lambda(\log P(f|+) - \log P(f|-)) \tag{3.3}$$

Where λ is the normalization constant to normalize $R(f)$ to [-1, 1]. The value of λ is determined by the range of $\log P(f|+) - \log P(f|-)$ across all features. Positive $R(f)$ shows that pattern f is good signal for

extracting closed-domain entities, and vice versa. Therefore we rewrite $T(e)$ as:

$$T(e) = \sum_{f} S(e, f)R(f) + c \tag{3.4}$$

Moreover, to ease sparsity in this phase, we applied iterative semi-supervised mechanism into the process. As shown in Algorithm 3.1 and Figure 3.2, we iteratively add top positive candidates and top negative candidates into the seed set. We re-calculate pattern scores after certain amount of iterations (here we chose 10 as our default iteration rounds) and use the new pattern scores to rank entities. Experimental results show that the semi-supervised mechanism works well for alleviating sparsity by adding more seeds and discover more patterns afterwards. M is chosen as 20 in our experiment.

Algorithm 3.1: Semi-NB for *Structure Creation*

```
Data: closed-domain corpus
Result:
Initialization;
for i = 0, i ⟨ iteration#, i++ do
    Calculate pattern score using seed set;
    Rank entities in candidate pool;
    Add top M positive entities into the pos seed set;
    Add top M negative entities into the neg seed set;
Calculate pattern score using updated seed set;
Smooth patterns using SPG;
Rank entities in candidate pool;
```

3.3.3 Semantic Pattern Graph

As discussed in Introduction, due to the lack of redundancy of entity mentions compared with web-scale data, *Dimension Creation* is especially challenging. To be more concrete, we show here the pattern scores we get without leveraging any public resources or smoothing in Figure 3.3. All the scores are normalized to scale [-1, 1]. On the left side, we show some selected pattern scores in the ranked list before leveraging Semantic Pattern Graph as a smoother. The numbers on the second column denote how much degree a pattern would tell a "good" closed-domain entity. Positive number means the pattern is a positive signal for closed-domain entity and vice versa. Clearly we can see three different types of patterns in Figure 3.3.

1. **Type I - Non-Sparse Good Patterns**: patterns like *"+develop"*, *"+infrastructure"* and *"−integrate"* belong to this category. These patterns are good indicators for chosen closed-domain entity dimension $\mathcal{L}$ and usually rank pretty high in the pattern list. They appears frequently with positive seeds and infrequently with negative seeds, and they normally have rich semantics strongly correlated to dimension

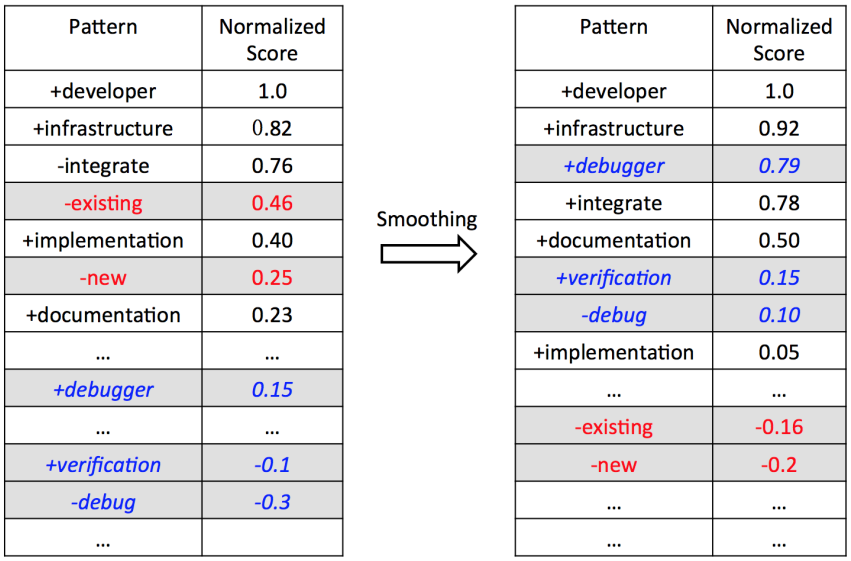


Figure 3.3: Contrast of scores before and after smoothing. Red/bold patterns are the mistakenly highly ranked general patterns, and blue/italic patterns are "good" sparse patterns.

$\mathcal{L}$;

2. **Type II - Bad General Patterns**: patterns like *"−existing"* and *"−new"* belong to this category. These patterns are bad indicators for $\mathcal{L}$ since they are too general. They usually do not have any semantics specifically correlated to dimension $\mathcal{L}$. They happened to appear with most positive seeds and not too many negative seeds, which leads to their high ranks in the pattern list. Normally if the positive seeds that user provided are much more popular than negative seeds (which is commonly the case), bad general patterns will contaminate the pattern list severely. The problem of these patterns being ranked high is that they may discover open-domain entities and make our entity set less accurate.

3. **Type III - Sparse Good Patterns**: patterns like *"+debugger"*, *"+verification"* and *"−debug"* belong to this category. These patterns have semantics strongly correlated to dimension $\mathcal{L}$ but they are not as commonly used as Type I patterns. They are good indicators for extracting closed-domain entities in dimension $\mathcal{L}$. However, different from Type I patterns, they stand for more specific/less common operators for dimension $\mathcal{L}$. In our example, every project can both be "debugged" and be "developed", but *"−develop"* is much more common. Therefore, Type III patterns usually can not rank very high and they may even possibly be on the negative side. Here, *"−debug"* has score -0.3 because it never appears with the pos seeds and appears with some neg seeds by accident.

Therefore, if we can demote Type II patterns and promote Type III patterns properly in the ranked pattern list, we can ease the trouble that sparsity brings and expect better quality of closed-domain entities.

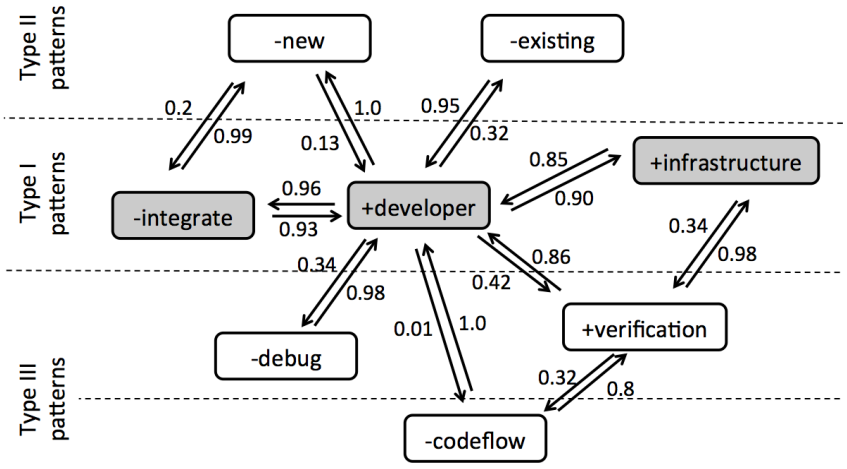


Figure 3.4: Toy Example of Semantic Pattern Graph from Microsoft dataset. For ease of exposition, only the edges between "+developer", "−integrate", "+infrastructure" and other patterns are shown in this example.

To do so, a deeper understanding of the semantic relations between patterns is required. However, it cannot be obtained from closed-domain corpus because of Sparsity again! The solution lies outside of the corpus: Web Data. It is possible to recover semantic structure from the web due to the fact that the patterns both closed-domain entities and public entities use are shared. Here we define the key concept **Semantic Pattern Graph** as follows:

Definition 3.1 ***Semantic Pattern Graph (SPG)****: A Semantic Pattern Graph is defined as a complete directed graph $G = (V, E)$, in which V is the pattern set and E contains edges representing relations between patterns. In **SPG**, every pair of distinct patterns is connected by a pair of edges.*

A sample **SPG** is shown in Figure 3.4. The scores on the edges are defined as follows:

$$S_{p\to q} = \frac{|E_p \cap E_q|}{|E_p|}, \quad S_{q\to p} = \frac{|E_p \cap E_q|}{|E_q|} \tag{3.5}$$

Here p, q are two distinct patterns. E_p, E_q denote the entity set that can appear with pattern p, q respectively in web data. If $S_{p\to q}$ is high, meaning that E_q is mostly covered by E_p, in other words, most entities that can appear with p, can also appear with q. Similarly, small $S_{p\to q}$ means that most entities that can appear with p, cannot appear with q. With such graph built, we can roughly divide semantic relations between patterns into three categories.

1. **Belonging:** If $S_{p\to q} \ll S_{q\to p}$, we say p is a general pattern of q. Semantically, q belongs to p. Examples are ⟨"−new", "+developer"⟩ and ⟨"+developer", "−debug"⟩, where "+developer" belongs

to "−new" and "−debug" belongs to "developer".

2. **Equal:** If $S_{p\to q} \approx S_{q\to p}$ and both of them are large, we say p, q are semantically equal since E_p, E_q share most entities. Examples includes ⟨"+developer", "−integrate"⟩ and ⟨"+developer", "+infrastructure"⟩

3. **Independent:** If $S_{p\to q} \approx S_{q\to p}$ and both of them are small, we say p, q are semantically independent, because E_p and E_q have very little overlap.

Revisit Figure 3.3 and Figure 3.4, we observe that Type I pattern are usually in the central layer of SPG, Type II patterns are roughly in top layer and Type III patterns are in the bottom layer. The layers in Semantic Pattern Graph are defined by how general the patterns are. Thus, lower layer entities usually form a **Belonging** relation to higher layer entities, while entities in the same layer normally form **Equal** or **Independent** relations. In other words, Type I patterns **belong** to Type II patterns and Type III patterns **belong** to Type I patterns. Another observation from Figure 3.3 is that Type I patterns are usually top patterns in the ranked pattern list before smoothing, since they obtains sufficient positive signals from closed-domain corpus. These observations inspire our design of smoothing algorithm.

In this section, we explain how we construct the **SPG** like Figure 3.4 from web data. We use Microsoft Web N-gram data, especially bi-gram sub-portion to construct **SPG**. Web Bi-gram dataset contains all possible combinations of patterns and their unigram entities on the web. To avoid unnecessary computation, we construct **SPG** solely based on the patterns appeared in our target corpus. Since web bi-gram data contains much noise, PMI between patterns and entities are used on bi-gram frequency to filter out insignificant bi-grams using a threshold. As an example, PMI of bi-gram "**P E**" will be calculated as:

$$pmi(\mathbf{P}, \mathbf{E}) = \log \frac{p(\mathbf{P}, \mathbf{E})}{p(\mathbf{P})p(\mathbf{E})} = \log \frac{count(\mathbf{PE})}{\sum count(\mathbf{P}*) \sum count(*\mathbf{E})} \tag{3.6}$$

Here **P** refers to the pattern and **E** refers to the entity. $count(\mathbf{PE})$ denotes the count of bi-gram "**P E**" on the web. $\sum count(\mathbf{P}*)$ denotes the total counts of all bi-grams including pattern **P** and $\sum count(*\mathbf{E})$ is defined similarly.

Anti Semantic Drifting Filter: Another problem we need to address is Semantic Drift of the same pattern on different domains. For example, **"−express"** may have specific meaning (genetically express the gene) in biology domain, while have no such semantics in any other field. That is to say, if we use the whole bi-gram data for constructing SPG for a specific domain, the semantic relations may be contaminated by semantics from other fields. As an example, if we see **"−express"** in computer science corpus, we need to avoid linking this one to **"−translate"**, a pattern that has semantic drift as well in biology domain. In this regard, we apply a filter on web "pattern-entity" pairs to preserve semantics of the target domain as much as

possible. We use the entity candidates generated in closed-domain corpus only to filter the "pattern-entity" pairs for building SPG. Thus in the **"−express"** case, semantics from biology domain will be filtered out while constructing SPGs for other fields.

After we filter out all the bi-grams containing those patterns and entities. We then calculate score E_p for each pattern p and construct the complete directed graph based on the statistics.

3.3.4 Smoothing Algorithm

In this section, we introduce the algorithm of using SPG for pattern re-evaluation. Back to our "projects" extraction task for Microsoft internal corpus. We observe in Figure 3.4 that Type I patterns like **"+developer"** and **"−integrate"**, are in the central layer of a the constructed SPG. Also, we observe that type II patterns, like **"−existing"** and **"−new"**, are more general than Type I patterns in the graph, indicating their **Belonging** relations. Type I patterns are more general than Type III patterns, like **"−debug"** and **"+verification"** in the graph, which also indicates their **Belonging** relations. Based on those observation, a possible solution is to locate the Type I pattern in the graph, and demote their general patterns (Type II) and promote their specific patterns (Type III). Since we also know that Type I patterns, with high possibility, will rank top in the pattern list, we then take the top-K patterns in the ranked pattern list, locate them in the **SPG** and smooth the rest of the graph using an induced graph from SPG. The induced graph is defined as follows:

Definition 3.2 ***Induced Smoothing Graph with Seeds (ISGwS)****: An Induced Smoothing Graph with Seeds is defined as a directed graph $G = (V, S, E)$, in which V is the pattern set, S is the seed set and E contains weighted edges representing smoothing score between patterns in S to patterns in V. Seed set S consists of top-K patterns in the ranked list before smoothing.*

As an example, Figure 3.5 shows the **ISGwS** generated from Figure 3.4. The new graph contains three seeds **"+developer"**, **"−integrate"** and **"+infrastructure"**. We evaluate how strong the score of a pattern should be smoothed by top-K patterns using formula as follows:

$$T_{p\to q} = \kappa \cdot (S_{q\to p} - S_{p\to q}) \cdot S_{p\to q} \cdot S_{q\to p} \tag{3.7}$$

Here p is the top pattern, q can be any other patterns in the graph. $\kappa = 4$ is the normalization constant to scale $T_{p\to q}$ to $[-1, 1]$. First term $(S_{q\to p} - S_{p\to q})$ in the formula determine the sign of $T_{p\to q}$. Moreover, smoothing effect grows as the difference between $S_{q\to p}$ and $S_{p\to q}$ grows. The second and third terms $S_{p\to q}, S_{q\to p}$ convey two things. 1). The bigger direction $S_{p\to q}$ (or $S_{q\to p}$, depends on demoting or promoting)

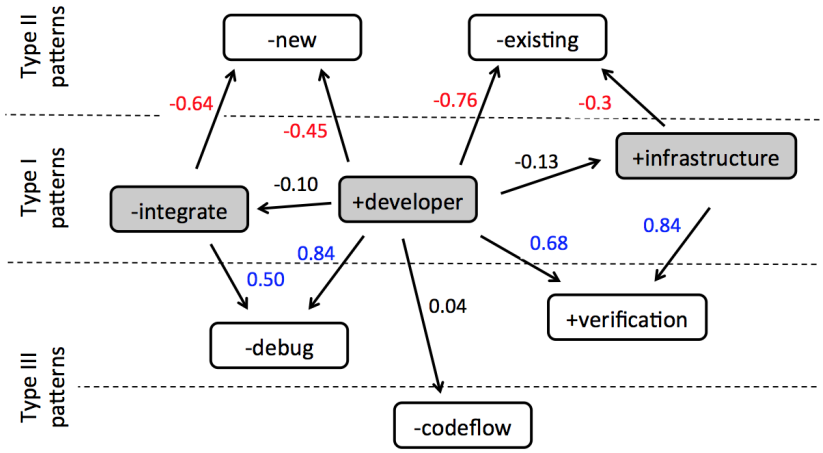


Figure 3.5: Toy Example of Induced Smoothing Graph with Seeds after scoring.

should be close to 1 to strongly show that there is a clear **Belonging** relation going on. For example, $S_{+developer \to -new}$ should be almost 1. 2). The smaller direction $S_{q \to p}$ (or $S_{p \to q}$) should also be big enough, such that the **Belonging** relation is not because of noise. For instance, pattern **"+codeflow"** in Figure 3.4 should not be promoted as much as **"−debug"** because that $S_{+developer \to +codeflow}$ is too small.

We get an Induced Smoothing Graph with Seeds by scoring the edges in Figure 3.4. The new graph is shown in Figure 3.5. By giving three top patterns, we strongly demote pattern "−new" and "−existing", promote pattern "−debug" and "+verification" and not do much to pattern "−integrate", "+infrastructure" and "+codeflow". The scoring approach satisfies our smoothing goal.

The last step is to average over all top-K patterns in the graph to smooth every other pattern, as follows:

$$R'(q) = (1 - \alpha)R(q) + \frac{\alpha}{K} \cdot \sum_{p \in top-K} T_{p \to q} \cdot R(p) \tag{3.8}$$

Here $R(q)$ and $R'(q)$ denote the pattern score before and after smoothing respectively. We use a linear combination of the original score and smoothed score to represent new score of pattern q. Moreover, we define the following form as *Smoothing Factor*:

$$SF(q) = \frac{1}{K} \sum_{p \in top-K} T_{p \to q} \cdot R(p) \tag{3.9}$$

After smoothing, we convert the pattern list in left side of Figure 3.3 to the right side list, as expected. A key problem here is how to choose α and K. α controls the balance between closed-domain signals and public signals. K controls the effective size of trustworthy patterns using only local signals. Thorough experiment

Method	P@50	P@100	P@200	P@300	P@500
Count	0.38	0.36	0.385	0.383	0.37
Capi	0.16	0.16	0.165	0.15	0.138
Hybrid	0.46	0.61	0.615	0.613	0.59
NB	0.74	0.71	0.72	0.687	0.66
SmooNB	**0.88**	0.81	0.82	0.76	0.744
SemiNB	0.84	0.77	0.78	0.753	0.712
Our System	0.86	**0.85**	**0.84**	**0.843**	**0.824**

Table 3.1: Precision at top K results for project/product extraction

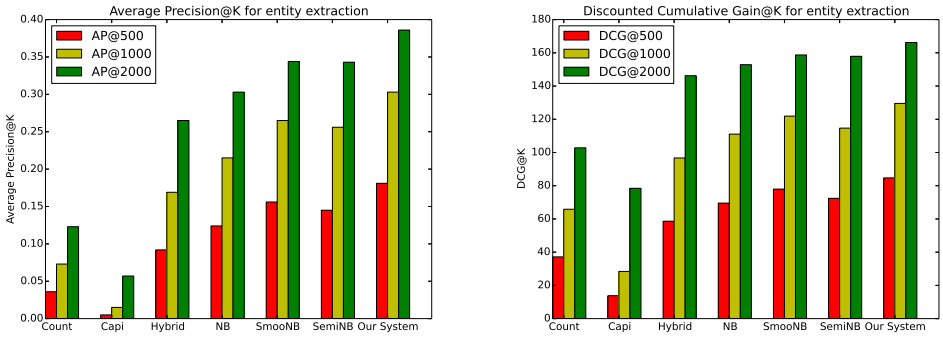


Figure 3.6: Average Precision and Discounted Cumulative Gain comparison

is conducted in Section 3.4 to answer this question.

3.4 Experimental Result

3.4.1 Enriching Microsoft Corpus

Since we model *Dimension-based Structure Creation* problem in a ranking framework, several ranking-based evaluation measures are proposed to assess both precision and recall of the result. Two intuitions are emphasized for the result we yield: 1) The 2,080 entities should rank high on the whole, 2) The top results are real entities with high probability. These intuitions align with the natural of Closed-domain Knowledge Base, that is with very high precision first being guaranteed, we then consider about coverage. Using this philosophy, we design three sets of evaluation measures as follows: a) Precision, evaluate the precision of top results from Prec@50 to Prec@500. b) Average Precision: AP@500, AP@1000, AP@2000 as

$$AP@n = \frac{\sum_{k=1}^{n} P(k)}{N} \tag{3.10}$$

where $P(k)$ means the precision at cut-off k in the entity list, N means the number of relevant entities in total, and c) Discounted Cumulative Gain: DCG@500, DCG@1000 and DCG@2000 as

$$DCG@n = \sum_{k=1}^{n} \frac{\mathbb{1}(e_k \in \mathbb{E})}{\log_2(k+1)} \tag{3.11}$$

Where $\mathbb{1}(e_k \in \mathbb{E})$ is the indicator function showing if k-th item is an closed-domain entity. Precision measurement emphasizes the top entity quality and AP/DCG measure the balance of both precision and recall. The result is shown in Table 3.1 and Figure 3.6.

From the result of Table 3.1, we observe that our system achieves above 80% precision even for top 500 results and the two baselines **Hybrid** and **NB** only achieves 59% and 66% respectively on top 500. To evaluate the Semi-supervised mechanism and Semantic Pattern Graph smoothing separately, we observe that both **SmooNB** and **SemiNB** can alleviate sparsity and enhance precision. It aligns with our intuition that Semi-supervised Learning can collect more patterns to capture entities with few mentions, while SPG leverage public signals to assign more accurate scores to each pattern. Table 3.1 also tells us that only applying smoothing can have better performance than only using semi-supervised learning. However, combing these two methods helps us to collect more low-precision patterns and assign them high confident scores, thus can yield best performance.

Figure 3.6 shows the performance on AP and DCG, with similar conclusion can be drew. AP and DCG are widely used to evaluate ranking qualities such as web search engines. We observe that using either Smoothing or Semi-learning outperforms the local ranker using only signals from corpus and positive seeds. Similar to precision, SPG smoothing improves more in terms of AP and DCG as well.

3.4.2 Pattern Study

In this section, we conduct experiments to look closer to patterns and answer two questions: 1) Why previous entity extractors with high-precision/low-recall patterns will fail in closed-domain setting? 2) How well the Semantic Pattern Graph performs to ease sparsity in a real case?

Pattern Coverage Study

To answer the first question, we implemented several pattern extractors used in previous systems: 1) NELL, 2) Knowitall and 3) SPIED. As discussed previously, NELL and SPIED used high-precision/low-recall patterns with part-of-speech (POS) restrictions to identify entities, while Knowitall uses mainly Hearst patterns and predefined templates for entity extraction. For Knowitall, we included all pattern templates

mentioned in their paper [21]. For NELL, we strictly follow their Coupled Pattern Learner [10] approach, where the patterns follows its POS rules and at most 100 instances/entities and 5 patterns are promoted. Patterns are ranked by precision and instances are filtered out unless the number of times it co-occurs with promoted pattern is at least three times more than the number of times it co-occurs with patterns left. For Spied, since the extracting details are not disclosed in the paper, we use the same POS restrictions that NELL used to extract patterns.

System	Knowitall	NELL	Spied	Our System
Coverage (%)	21.5	54.4	54.3	**81.7**

Table 3.2: Coverage of projects/products using extracted lexical patterns

Table 3.2 shows the coverage of the 2,080 labeled projects using different lexical pattern set generated by four systems. We observe that our system can potentially extract more than 80% of target closed-domain entities, while in the meantime, NELL/Spied/Knowitall can only find less than 55% entities. That is to say, in a sparse setting, even the best existing classifier/ranker would lose about half of the entities. Knowitall has the lowest coverage due to the limited pattern templates it uses. Hearst patterns have poor coverage in closed-domain corpus. NELL and Spied use more general patterns confined by specific POS templates can yield better entity coverage than Knowitall, but are still deficient for a sparse setting. Because the POS constraints normally require verbs in the pattern or require explicit relations to another noun phrases. However, many closed-domain entities lack of the redundancy of such patterns. For example in our system, two patterns "+platform" and "−native" are good signals for identifying closed-domain entities, but will not be considered by precious web-scale extractors. Note that 20% of the entities in Microsoft are still unreachable by our patterns, some of them are never mentioned in the corpus and some of them are mentioned only a few times with non-informative patterns (the general patterns which are not exclusive to projects/products).

A reasonable concern of introducing less restricted patterns is that it potentially lower our confidence on judging every single pattern. Hence we introduced smoothing based on Semantic Pattern Graph to get accurate evaluation of each pattern.

3.5 Summary

This chapter introduced our framework for *Dimension-based Structure Creation*, which involves semi-supervised bootstrapping entity extraction framework using seed sets. It is the first work to introduce the concept of Semantic Pattern Graph into entity extraction problem and model the semantics between contextual patterns to enhance entity coverage and precision. With detailed analysis, we show that in a

Pattern	Score Before	Smo.	Score After	#Mention	Examples
+debugger	0.157	0.895	0.378	136	... from an attached **VS10** *debugger*. ...triggered the **jit** *debugger* dialog...
+backend	0.647	0.983	0.748	108	Both the **SQL Cluster** *Backend* and... ...instance of **FAST Search Server** *backend*...
+verification	-0.094	0.704	0.145	77	Failed **authenticode** *verification* of payload.
-existing	0.456	-0.532	0.160	1789	...migrate the *existing* **ORANGE** connection.. ..want all *existing* **allocations** such for...
-new	0.253	-0.995	-0.121	8239	...to create *new* **experience** I get... ...in *new* **Software Distribution** policies...

Table 3.3: Smoothed patterns along with the score before smoothing, smoothing factor and score after smoothing, and the example entities they appear with. Scores in columns 2-4 are normalized to [-1, 1], where positive score means it is good pattern for extracting product/project, and vice versa. Score After is the linear combination of Score Before and Smoothing Factor using $\alpha = 0.3$. Mention Count indicates how popular the pattern is in the corpus. Entity examples for the first three patterns are good closed-domain entities which are promoted due to the promotion of their patterns, and the entities for last two patterns are the mistakenly classified entities which are demoted due to the demotion of their patterns.

sparse setting, leveraging public web-scale data to understand semantic relations between lexical patterns can effectively alleviate the sparsity. Hence, many entities with very little evidence can be possibly correctly extracted. In the experiment part, we also show that the smoothing algorithm using SPG can significantly improve precision and recall of learned closed-domain entities. We also conduct several other experiments to show our smoothed extractor outperforms methods without understanding pattern relations.

The proposed Semantic Pattern Graph give a unique perspective of relations between patterns (instead of entities). By leveraging the relations obtained from abundant public signal, the closed-domain data sparsity problem can be largely eased. Therefore, the proposed framework in this chapter is general enough to be applied to other Information Extraction tasks on closed-domain corpus.

Chapter 4

Cell-based Document Allocation via Dimension-aware Joint Embedding

4.1 Overview

Multi-dimensional cell-based document allocation is closely related to text categorization [1], yet there are notable differences between the two problems. Text categorization is typically formulated as a classification problem: assuming a corpus of training documents is given, the process is to extract different features for those documents and learn a classifier that can infer the ground-truth labels for any documents. Cell-based document allocation, on the other hand, does not have labeled training documents in most scenarios. More importantly, it is a multi-dimensional categorization problem in nature. For example, Figure 1.1 shows an illustrating task of news document allocation. As shown, to allocate each news article into a cube cell (*i.e.*, a multi-dimensional label like $\langle$*Sports, 2017, USA*$\rangle$), we need to consider different dimensions simultaneously: 1) the topic dimension: what is the event about? 2) the time dimension: when does the event happen? and 3) the location dimension: in which country does the event happen? To neatly organize such a news corpus, it is necessary to perform multi-dimensional categorization for each article and selects an appropriate label from every cube dimension.

Existing text categorization methods, due to the fact that they ignore the dependencies among different dimensions in a text cube, face nontrivial difficulties for cell-based document allocation. The first one is the scarcity of labeled training data. The success of existing document classification models is largely relying on sufficient labeled data to train reliable document classifiers. However, it is often costly to obtain enough training data for the document allocation problem. Indeed, as every training document has to be assigned with one or several labels from each dimension, the labeling process is too costly to realize for cell-based document allocation.

The second difficulty is to extract features that are discriminative for all the cube dimensions. The feature extraction procedure for existing text categorization methods falls into two classes: 1) extracting hand-crafted features (bag-or-words, tri-letter, *etc.*) for documents; and 2) learning distributed representations for textual units (words, sentences, or documents) as document features. Both strategies generate a fixed feature representation for each document. However, different cube dimensions often require different

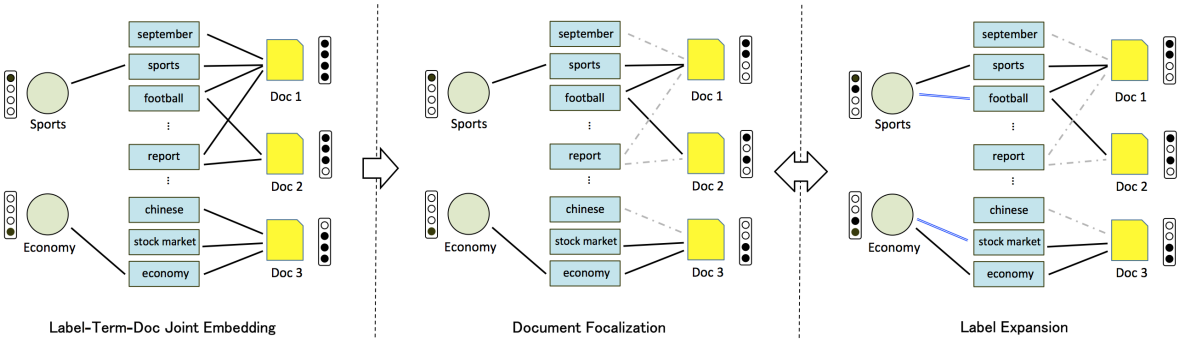


Figure 4.1: A toy example of dimension-aware joint embedding framework on the *topic* dimension. In document focalization, the background term ("report") along with the indiscriminative words ("september" and "chinese") are less emphasized for the *topic* dimension. In label expansion, more topic-indicative words ("football" and "stock market") are expanded and labeled.

representations for the same document. For instance, in the above news article categorization problem shown in Figure 1.1, the location dimension may favor a feature set that captures location-related words such as "Chicago" and "China", while the topic dimension favors a feature set that emphasizes semantics-telling terms such as "super bowl" and "manufacturing industry". However, existing text classification methods are dimension-agnostic, their derived fixed feature representations often incorporate irrelevant information for the considered dimension and may not work well for all the cube dimensions.

In this chapter, we propose a *dimension-aware joint embedding* method for effective cell-based document allocation. Our method does not require any costly human labeling process. Instead, it simply uses the label names in different cube dimensions as a small set of labeled seed terms. Regarding these seed terms as weak supervision, we develop a dimension-aware joint embedding framework that learns different embedding spaces for different cube dimensions. As a result, each document can have multiple representations, each tailored for one cube dimension to achieve high discriminative power.

Our idea to obtain dimension-aware document representations is to propagate the label information from seed terms. Starting with static term embeddings, we iteratively estimate the distinctiveness scores of the terms for each cube dimension, and use such scores to derive dimension-aware document embeddings. As shown in Figure 4.1, we construct a tripartite graph that captures the correlations among labels, terms, and documents, and progressively update the graph to obtain dimension-aware representations. This process is called *document focalization*. Simultaneously, the labels are connected to more terms that are highly discriminative to them. This process is called *label expansion*. Such an updating process yields two advantages for the allocation problem. First, the label-term subgraph is gradually densified to propagate label information from the seed terms to semantically similar terms. Consequently, more labeled terms are generated to allevi-

ate the label scarcity problem of the initially chosen seeds. Second, the term-document sub-graph gradually focuses on those discriminative terms. Discriminative dimension-aware representations for the documents are generated and tailored for the target cube dimension.

Our contributions can be summarized as follows:

1. We propose a label-efficient framework for cell-based document allocation. It does not require excessive labeled data, but simply leverages the surface names of different labels to achieve effective document categorization along all the cube dimensions.

2. We propose a novel dimension-aware joint embedding algorithm. It learns dimension-driven representations for documents by iteratively refining the distinctiveness scores of different terms, which leads to more discriminative document representations compared with previous task-agnostic feature learning methods.

3. We have performed extensive experiments using real data sets. The results show that our method generates high-quality document representations and improves the allocation accuracy of state-of-the-art methods ranging from *dataless classification* [55] to *topic modeling* [31].

This chapter studies the problem of cell-based document allocation. In tradition data cubes [26, 12], the allocation process is termed as *cube instantiation* or *cube loading*. Here we adopt the term *cell-based document allocation* to refer to the process of organizing documents into the multi-dimensional space. In this part, we assume the dimension structure has already been defined with domain knowledge and focus on the document allocation problem. We formally define this problem in the following.

Problem 4.1 (Multi-dimensional Cell-based Document Allocation) *Let $\mathcal{TC}$ be a text cube with dimensions $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$. For any document d in a corpus $\mathcal{D}$, the cell-based document allocation problem is to assign n labels $l_{t_1}, \ldots, l_{t_n}$ for d, where label $l_{t_i} \in \mathcal{L}_i$ represents the category of d in dimension $\mathcal{L}_i$.*

We propose a dimension-aware joint embedding method for effective allocation. Different from existing supervised text categorization techniques, our method uses the surface label names from each dimension as weak supervision to form a small set of seed terms. Based on such labeled terms and the label-term-graph correlations, it then learns dimension-aware document representations for categorization. In specific, our method has the following major steps for realizing cell-based document allocation:

1. **Joint Label-Term-Document Embedding**: Based on the correlations among labels, seed terms, and documents, we embed different labels, terms, and documents into a joint space via a graph-based approach.

Table 4.1: Notations used in this chapter.

Notation	Meaning
$\mathcal{D}, \mathcal{T}$	the set of documents and the set of extracted terms
$\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_n$	the n categorical dimensions in the text cube
$\mathbb{U}^{\mathcal{L}}, \mathbb{U}^{\mathcal{T}}, \mathbb{U}^{\mathcal{D}}$	the learned embedding matrices for labels, terms and documents
$u_i^{\mathcal{L}}, u_j^{\mathcal{T}}, u_k^{\mathcal{D}}$	the embedding vectors of i-th label, j-th term and k-th document
G_{LTD}	the constructed Label-Term-Document graph
$\mathbf{A}^{(\mathcal{LT})}$, $\mathbf{A}^{(\mathcal{TD})}$	the adjacency matrix of label-term subgraph and term-document subgraph
$\mathbf{R}^{(\mathcal{DL})}$	the document-label similarity matrix, $\mathbf{R}_{i,j}^{(\mathcal{DL})}$ is the similarity between d_i and l_j
$\mathbf{R}^{(\mathcal{TL})}$	the term-label distribution matrix, row i is the label distribution of t_i
$f(t_i, \mathcal{L}), f(t_i, l_j)$	the dimension-focal score of term t_i on dimension $\mathcal{L}$ and label-focal score of term t_i on label l_j
$e(t_i, l_j)$	the expansion score of expanding term t_i onto label l_j

2. **Dimension-Aware Embedding Updating**: With the learned joint label-term-document embeddings, we derive dimension-aware document embeddings by focusing on discriminative terms for each dimension; and also expand the seed terms to address label sparsity.

3. **Embedding-Based Label Allocation**: Finally, we assign a label to each document using the learned dimension-aware document embeddings and seed-expanded label embeddings.

Table 4.1 shows the notations used in this section and the rest of this chapter.

4.2 Joint Label-Term-Document Embedding

In this section, we describe the joint label-term-document embedding step. For a given dimension $\mathcal{L}$, it first constructs a *Label-Term-Document* tripartite graph G_{LTD}. (Sec. 4.2.1) to encode the co-occurrence relationships between labels, terms and documents in the corpus; and then embeds different data types into the same space (Sec. 4.2.2).

4.2.1 Label-Term-Document Graph

To allocate documents into proper cube cells, we choose to learn latent embeddings for documents and labels instead of using the bag-of-words representation, due to the fact that embeddings alleviate the term sparsity problem by discovering the correlations among terms. To model labels, terms and documents in the same embedding space, we propose to construct a *Label-Term-Document (L-T-D)* graph to encode their co-occurrence relationships.

Since different dimensions have different label spaces, we construct an L-T-D graph for each dimension, and learn joint-embeddings for different dimensions separately. As shown in Figure 4.1, in an L-T-D graph, there are three different node types for labels, terms, and documents, respectively. Built upon these node

types, we induce two different edge types to encode the co-occurrence information in the input corpus: (1) label-term edges; and (2) document-term edges. The resultant L-T-D graph is a heterogeneous tripartite graph defined as follows.

Definition 4.1 (L-T-D Graph) *An L-T-D graph, denoted as* $G_{LTD} = (\mathcal{L} \cup \mathcal{T} \cup \mathcal{D}, E_{TL} \cup E_{TD})$*, is a tripartite graph: (1)* E_{TL} *is a set of edges between labels and terms. There is an edge between term* t_i *and label* l_j *if and only if they strictly match each other, and the weight* $w_{i,j}^{TL}$ *is set to* 1*; (2)* E_{TD} *is a set of edges between terms and documents. There is an edge between term* t_i *and document* d_j *if* t_i *occurs in* d_j*, and the edge weight weight* $w_{i,j}^{TD}$ *is set to* $\log(1 + count(t_i, d_j))$*.*

4.2.2 Graph Embedding

In the same vein as other graph embedding algorithms [58, 57, 27], we learn an embedding vector for each node in $\mathcal{L} \cup \mathcal{T} \cup \mathcal{D}$ by collectively embedding the two bipartite networks. This is achieved by minimizing the following objective function:

$$\mathcal{O}_{ltd} = -\sum_{(i,j)\in E_{TL}} w_{i,j}^{TL} \log p(u_i^T|u_j^L) - \sum_{(i,j)\in E_{TD}} w_{i,j}^{TD} \log p(u_i^T|u_j^D) \tag{4.1}$$

where

$$p(u_i^T|u_j^L) = \frac{exp(u_i^T \cdot u_j^L)}{\sum_{i'\in\mathcal{T}} exp(u_{i'}^T \cdot u_j^L)} \tag{4.2}$$

$$p(u_i^T|u_j^D) = \frac{exp(u_i^T \cdot u_j^D)}{\sum_{i'\in\mathcal{T}} exp(u_{i'}^T \cdot u_j^D)} \tag{4.3}$$

The first term in (4.1) is the objective function to embed label-term subgraph, whereas the second term is the objective function to embed term-document subgraph. For each label l_j, Eq (4.2) defines the conditional distribution of $p(\cdot|u_j^L)$ over all the terms in set $\mathcal{T}$. Ditto for Eq (4.3).

The objective function (4.1) is expensive to optimize due the large amount of terms in the graph. We use edge sampling and negative sampling [43] with stochastic gradient descent to optimize. When sampling an edge $e = (i, j)$ in either E_{TL} or E_{TD}, multiple negative edges are sampled from a noise distribution. Since we have two edge types in (4.1), two sampling strategies can be implemented: 1) iteratively sample $e_1 \in E_{TL}$ and $e_2 \in E_{TD}$, 2) sample E_{TD} until convergent and then sample E_{TL} until convergence. In our experiments, we observe the first policy achieves better representations.

We call the above process *Joint Embedding Learning*. In the end, term embeddings $\mathbb{U}^{\mathcal{T}}$, document embeddings $\mathbb{U}^{\mathcal{D}}$ and label embeddings $\mathbb{U}^{\mathcal{L}}$ are obtained to encode their semantics in a latent space.

4.3 Dimension-Aware Embedding Updating

In this section, we present the dimension-aware embedding step. Taking the joint embeddings as initialization, the updating step iteratively derives dimension-aware document embeddings by focusing on discriminative terms for each dimension, and meanwhile expands the initial labeled seed terms to alleviate the label sparsity problem. In the following, we first introduce the idea of dimension-aware focalization and document embedding, and then describe how we perform label expansion.

4.3.1 Focal Score for Discriminative Power

Although the joint embeddings capture the co-occurrence information among labels, terms, and documents, the resultant embeddings suffer from two problems. First, the scarcity of labeled terms makes the label embeddings not comprehensive enough to cover the semantics of the target category. For example, with our weak supervision scheme, the label *Sports* is linked to the term *sports*. However, the scope of *Sports* is quite broad, covering information such as *nba*, *nfl*, *soccer*, *etc.*On the other hand, the document embedding is fixed for all dimensions. In cell-based document allocation, different dimensions requires different representation for the same document. For instance, the location dimension may favor terms that captures location-related information, such as *new york*, while topic dimension may favor terms that captures topical information, such as *super bowl* and *economic growth*. Consequently, the initial joint embeddings *under-represent* labels while over-represent documents.

To tackle the above two problems, the key is to estimate the terms' discriminative power *w.r.t.* a dimension, as well as *w.r.t.* a label. For the over-represented document embeddings, the information from background and other dimensions can be removed if we know how discriminative each term is. For *under-represented* label embeddings, highly relevant information can be expanded if we know the most discriminative terms *w.r.t.* the target label. Therefore, we propose *dimension-focal score* and *label-focal score*.

Definition 4.2 (Dimension-Focal Score) *The dimension-focal score* $f(t, \mathcal{L}_i) \in [0, 1]$ *of a term* t *is its discriminative power* w.r.t. *dimension* $\mathcal{L}_i$. *The higher* $f(t, \mathcal{L}_i)$ *is, the more discriminative term* t *is for deciding the label in* $\mathcal{L}_i$.

Definition 4.3 (Label-Focal Score) *The label-focal score* $f(t, l) \in [0, 1]$ *of a term* t *is its discriminative measure* w.r.t. *the label* l *in dimension* $\mathcal{L}$. *The higher* $f(t, l)$ *is, the more exclusively term* t *belongs to label* l.

The focal scores indicates the importance of a term in assigning a particular dimension. We denote the adjacency matrix form of E_{TL} and E_{TD} as $\mathbf{A}^{(\mathcal{LT})}$ and $\mathbf{A}^{(\mathcal{TD})}$. To compute focal scores, we define the term-label distribution matrix as follows:

$$\mathbf{R}^{(\mathcal{TL})} = \mathbf{A}^{(\mathcal{TD})}\mathbf{R}^{(\mathcal{DL})}, \tag{4.4}$$

where $\mathbf{R}^{(\mathcal{DL})}$ is the label-document similarity matrix given by:

$$\mathbf{R}^{(\mathcal{DL})} = \mathbb{U}^{\mathcal{D}}\mathbb{U}^{\mathcal{L}^T}. \tag{4.5}$$

In the above equations, $\mathbf{R}^{(\mathcal{DL})}$ calculates the similarity between documents and labels in the embedding space. Thus Equa 4.4 use the term-doc adjacency matrix to indirectly compute the label distribution of each term. Base on the resulting similarity matrix $\mathbf{R}^{(\mathcal{TL})}$, we apply row-wise softmax function to derive the probability distribution of each term over the labels. After applying the softmax function, we define the *label-focal score* $f(t_i, l_j)$ as the probability of assigning term t_i to label l_j. Namely,

$$f(t_i, l_j) = \mathbf{R}_{ij}^{(\mathcal{TL})} \tag{4.6}$$

With the term-label distribution matrix, we further compute the *dimension-focal score* of each term using its label distribution's normalized KL-divergence from uniform distribution. We compute dimension-focal score of term t_i as:

$$f(t_i, \mathcal{L}) = \frac{\sum_{j=0,\cdots,|\mathcal{L}|} \mathbf{R}_{ij}^{(\mathcal{TL})} \log |\mathcal{L}| \mathbf{R}_{ij}^{(\mathcal{TL})}}{\log |\mathcal{L}|} \tag{4.7}$$

where $\log |\mathcal{L}|$ is a normalization term.

The resulting focal scores are within range $[0, 1]$. Note that the dimension-focal score of a term is dimension-dependent. A term with high dimension-focal score in one dimension may have very low dimension-focal scores in other dimensions, depending on how important the term is to a dimension.

4.3.2 Document Focalization

We develop *Document Focalization* to solve the *over-represented* problem for document embeddings. The intuition is that the fixed document representation embeds information related to multiple dimensions. As shown in Figure 4.1, the documents are connected to topical words as *football* and *stock market*, location-related words as *chinese*, time-related words as *september* and background terms as *report*. Without focusing

on the highly relevant terms, the other terms consisting the document will act as background noise and make classification difficult. The process of document focalization, denoises irrelevant information by focalizing document representations (*e.g.*, lower the weights of *chinese*, *september* and *report* in Figure 4.1). We propose the dimension-dependent embedding for document d_i as follows:

$$\mathbb{U}^{\mathcal{D}} = \left(\mathbf{A}^{(\mathcal{TD})} \circ \left[f_{\mathcal{L}} \cdots f_{\mathcal{L}} \right]_{|\mathcal{T}| \times |\mathcal{D}|} \right)^{T} \mathbb{U}^{\mathcal{T}} \tag{4.8}$$

where $\circ$ is the Hadamard product (*i.e.*, entrywise product) and $f_{\mathcal{L}}$ is the dimension-focal score vector *w.r.t.* dimension $\mathcal{L}$ on all terms; and $|\mathcal{T}|$ is the length of $f_{\mathcal{L}}$. In this formula, the dimension-focal score is a $[0, 1]$ penalty on the original w^{TD} weight. The document embedding is the aggregation of its term embeddings with penalized weights. The higher the term's dimension-focal score is, the more it is emphasized when computing document embeddings.

When computing $\mathbf{R}^{(\mathcal{DL})}$, the doc-label similarity matrix is computed using document embedding $\mathbb{U}^{\mathcal{D}}$. Thus the dimension-focal scores of terms are updated as long as $\mathbb{U}^{\mathcal{D}}$ is updated. Therefore, the updating of $f_{\mathcal{L}}$ and $\mathbb{U}^{\mathcal{D}}$ is dependent on each other. According to Equation 4.4, 4.6 and 4.8, we design an iterative process as shown in Algorithm 4.1. In a nutshell, based on the initial $\mathbb{U}^{\mathcal{D}}$, we iteratively update $\mathbf{R}^{(\mathcal{DL})}$, $f_{\mathcal{L}}$ and $\mathbb{U}^{\mathcal{D}}$ until they stabilize.

4.3.3 Label Expansion

We develop *Label Expansion* to solve the *under-represented* problem for label embeddings. The intuition behind it is straightforward. To link the label with more discriminative terms, *e.g.*, term *football* to label *Sports* and term *stock market* to label *Economy* in Figure 4.1, the enriched label representation can match more relevant documents. Therefore, the expansion of terms with high label-focal score can benefit the label representation.

To ensure the quality of the expanded terms, we also consider the popularity of a term. Since many terms that have high discriminative power are infrequent in the corpus, expanding them not only covers few extra documents, but also suffers from their inadequatly-trained embeddings. Hence, we design the expansion criteria as a combination of focal score and popularity:

$$e(t_i, l_j) = f(t_i, l_j) \cdot \frac{\log 1 + df(t_i)}{\log 1 + |\mathcal{D}|} > \eta \tag{4.9}$$

where the second term is the normalized term popularity and $df(t_i)$ is the document frequency of term t_i. We set η as the threshold. Any term-label pair with the expansion score higher than η is connected and

the adjacency matrix $\mathbf{A}^{(\mathcal{LT})}$ is updated. After the expansion, similar to document embedding, the label embedding is updated as:

$$\mathbb{U}^{\mathcal{L}} = \mathbf{A}^{(\mathcal{LT})}\mathbb{U}^{\mathcal{T}} \tag{4.10}$$

Since the label expansion process changes the label embeddings, the focal scores will be updated according to the new $\mathbf{R}^{(\mathcal{DL})}, \mathbf{R}^{(\mathcal{TL})}$. Similarly, after the one round of document focalization, both document embedding and focal scores are updated, the new label expansion is required. Therefore, we design an interactive process to apply label expansion and document focalization in turns. The entire algorithm of representation updating is detailed in Algorithm 4.1.

Algorithm 4.1: Representation Updating Algorithm

$\mathbb{U}^{\mathcal{L}}$, $\mathbb{U}^{\mathcal{D}}$, $\mathbb{U}^{\mathcal{T}}$: Initial embeddings of labels, documents and terms.
$\mathbf{A}^{(\mathcal{LT})}$, $\mathbf{A}^{(\mathcal{DL})}$: adjacency matrix in the L-T-D graph.
K_1, K_2: outer iteration counter and inner iteration counter
for $i = 1, \ldots, K_1$ **do**
    `// Document Focalization`
    for $j = 1, \ldots, K_2$ **do**
        compute $\mathbf{R}^{(\mathcal{TL})}$ by 4.4 and 4.6
        for t_i *in* $\mathcal{T}$ **do**
            $f(t_i, \mathcal{L}) = \frac{\sum_{j=0,\cdots,|\mathcal{L}|} \mathbf{R}_{ij}^{(\mathcal{TL})} \log |\mathcal{L}| \mathbf{R}_{ij}^{(\mathcal{TL})}}{\log |\mathcal{L}|}$
        $\mathbb{U}^{\mathcal{D}} = \left(\mathbf{A}^{(\mathcal{TD})} \circ \left[f_{\mathcal{L}} \cdots f_{\mathcal{L}}\right]_{|\mathcal{T}|\times|\mathcal{D}|}\right)^{T} \mathbb{U}^{\mathcal{T}}$ `// Update document embedding`
    `// Label Expansion`
    compute $e(t, l)$ for all term-label pairs by (4.9)
    update $\mathbf{A}^{(\mathcal{LT})}$ for all $e(t, l) > \eta$
    $\mathbb{U}^{\mathcal{L}} = \mathbf{A}^{(\mathcal{LT})}\mathbb{U}^{\mathcal{T}}$ `// Update label embedding`

4.4 The Overall Algorithm

In this section, we present embedding-based label allocation step and then summarize the overall allocation process. The label allocation step is achieved by directly measuring the cosine similarity between label embedding and document embedding. Namely

$$label(d_i) = \underset{l_j \in \mathcal{L}}{\operatorname{argmax}} \cos(u_i^D, u_j^L) \tag{4.11}$$

Based on the joint embedding step, the dimension-aware updating, and the above label allocation step, we summarize the cell-based document allocation algorithm in Algorithm 4.2. As shown, given the corpus,

we first build the L-T-D tripartite graph and compute the joint embeddings of labels, terms, and documents. Then we iteratively update the embeddings based on Algorithm 4.1 to derive dimension-aware document and label embeddings. Finally, we assign the max-scoring label to each document for the target dimension.

Algorithm 4.2: Dimension-aware Joint-Embedding for Cell-based Document Allocation

$\mathcal{D}, \mathcal{T}$: the documents and extracted terms
$\mathcal{L}_1, \cdots, \mathcal{L}_n$: the label sets for n dimensions
M, K: number of sampled edges and number of negative samples
for $\mathcal{L}$ *in* $\mathcal{L}_1, \cdots, \mathcal{L}_n$ **do**
    construct G_{LTD} using $\mathcal{D}, \mathcal{T}, \mathcal{L}$
    `// Embedding learning`
    randomly initialize $\mathbb{U}^{\mathcal{L}}$, $\mathbb{U}^{\mathcal{T}}$ and $\mathbb{U}^{\mathcal{D}}$
    while *iter* ¡ M **do**
        sample an edge $e \in E_{TL}$ and K negative edges
        update $\mathbb{U}^{\mathcal{T}}$ and $\mathbb{U}^{\mathcal{L}}$
        sample an edge $e \in E_{TD}$ and K negative edges
        update $\mathbb{U}^{\mathcal{T}}$ and $\mathbb{U}^{\mathcal{D}}$
    `// Embedding updating`
    $\mathbb{U}^{\mathcal{D}}$, $\mathbb{U}^{\mathcal{L}}$ = Embed_Update(G_{LTD}, $\mathbb{U}^{\mathcal{L}}$, $\mathbb{U}^{\mathcal{D}}$, $\mathbb{U}^{\mathcal{T}}$)
    `// Construction using embeddings`
    for d_i *in* $\mathcal{D}$ **do**
        label(d_i) = $\text{argmax}_{l_j \in \mathcal{L}} \cos(u_i^D, u_j^L)$

4.5 Experiments

4.5.1 Experimental Setup

Dataset

To evaluate our multi-dimensional document allocation, we collect a corpus of New York Times articles from 2015. 13,080 articles are crawled and using New York Time API[1]. The collection contains news covering 29 topics and 14 countries. Thus two dimensions, *Topic* and *Location*, are used in our experiments. Each article has a topic label and a country label that are given by data providers. We use these annotations as ground truth. An illustration of the labels in the two dimensions are given in Figure 4.2. To build the L-T-D graph, we use an existing phrase mining tool [2] to extract the high-quality phrases and then remove stopwords.

[1] http://developer.nytimes.com/
[2] https://github.com/shangjingbo1226/SegPhrase

Topic (29): Federal Budget, Surveillance, Arts, Dance, Football, Sports, Stocks and Bonds, Immigration, Cosmos, Television, Law Enforcement, Energy Companies, Hockey, Business, Basketball, Gay Rights, Science, Tennis, Golf, Politics, Music, Economy, Environment, Movies, Gun Control, Baseball, Soccer, International Business, The Affordable Care Act, Military, Abortion
Location (14): USA, China, UK, France, Germany, Belgium, Ireland, Australia, Netherlands, Russia, Spain, Canada, Brazil, Italy

Figure 4.2: Dimension labels in NYT dataset (number of labels given in parenthesis)

Baselines

To demonstrate the effectiveness of our method, we compare it with multiple baselines. The baselines are implemented by treating each dimension as an independent categorization task.

1. **IR**: This method treats each label as a search query. The document relevance is measured by *TF-IDF* score. For a particular document, the label is assigned to the query with highest relevance score.

2. **IR + Query Expansion (IR+QE)**: In this method, the queries derived from label names are expanded using *word2vec* similarity [51, 17]. Each query is expanded by 3 most similar terms. The relevance score is the averaged *TF-IDF* score. This baseline shares similar idea of label expansion.

3. **Word2vec**: Word2vec is popular in learning representation of words. We first learn 100-dim word representations. The label and document representations are aggregated using the contained words. The label of a document is the one with highest cosine similarity.

4. **Word2vec + Document Focalization (Word2vec+DF)**: Instead of simply aggregating word representation for document representation, we leverage *Dimension-Focal Score* of each word as in our approach. The document representation is then aggregated with *Document-Focal Score* as the weight.

5. **Topic Model (TM)**: Traditional topic model [7] does not take label names as input. Nevertheless, we train a topic model with 50 topics and assign each topic to a label in each dimension. The allocation is based on the topic distribution of label names.

6. **Topic Model with Prior (TMwP)**: Following [39], we use a semi-supervised topic modeling, specifically PLSA with label names as prior, to categorize each document into a label. The allocation is based on the topic distribution of the document.

   Traditional topic model [7] does not take label names as input. Nevertheless, we train a topic model with 50 topics and assign each topic to a label in each dimension. The allocation is based on the topic distribution of label names.

7. **Dataless Classification**: Dataless [55, 11, 13] is an unsupervised algorithm that utilizes Wikipedia as external knowledge base. In our experiment, we pick 500 wiki concepts for each label. The document label is determined by its similarity to the expanded Wikipedia pages.

Besides baselines, we also design three ablation algorithms to evaluate the separate effects of *Joint Embedding*, *Document Focalization* and *Label Expansion*.

1. **Joint Embedding (JE)**: We directly use the embeddings of labels and documents after joint embedding.

2. **Joint Embedding + Document Focalization (JE+DF)**: The document embedding is updated for each dimension using *Focal Score*. The label embedding still uses the original embedding learned from joint embedding.

3. **Joint Embedding + Label Expansion (JE+LE)**: The label embedding is updated by the expanded seeds. The document embedding is from direct joint embedding.

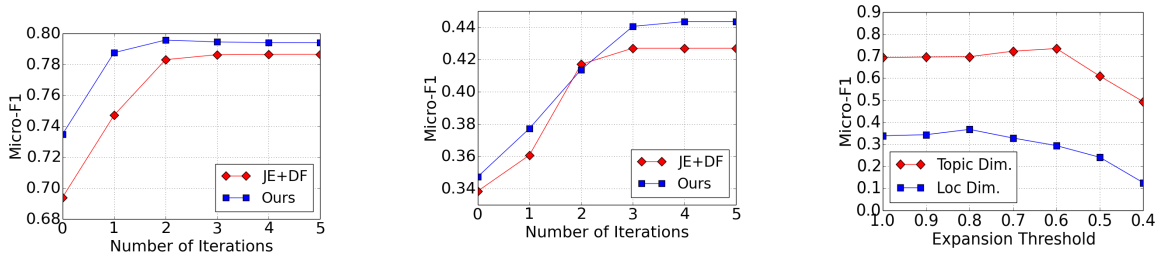
Evaluation Metric

We use averaged F_1 scores to measure the performance of all the methods. In particular, both micro-averaged and macro-averaged F_1 are used. Macro-F_1 focus more on the individual labels, such that labels with fewer documents are equally important as the popular labels. Micro-F_1 measures the performance across all labels, thus the popular labels weigh more in the final F_1 score.

4.5.2 Effectiveness Evaluation

Overall Performance with NYT Cube

We first evaluate the overall performance of 7 baselines, 3 ablations and our method. The result is shown in Table 4.2. The proposed Dimension-aware Joint Embedding method outperforms the baseline methods and ablations on both the *Topic* dimension and the *Location* dimension. It is observed that the performance on the *Topic* dimension is better than that on *Location* dimension for almost every method. That is because the topic in news articles are the primary dimension. The majority of the content is about the topic of the event, while a smaller portion of words indicates the location of the event. However, for methods that use dimension-agnostic document representations, *e.g.*, Word2vec and Topic Model, the performance drop more than methods that use dimension-aware representations, *e.g.*IR, Dataless, TMwP and our method. This validates our design that different dimensions require different representations for the same document.



(a) Topic Dimensiom - DF

(b) Location Dimension - DF

(c) Label Expansion Performance

Figure 4.3: Performance change as *Document Focalization* and *Label Expansion* iterates.

To evaluate the *Document Focalization* module, we observe clear improvements on Joint-Emb and Word2vec after applying document focalization. However, the improvement on Joint-Emb is much higher. That is because document focalization is the process of finding terms that can discriminate documents from different labels, while word2vec embeds the context window without considering the document-level co-occurrence. Thus the *Dimension-Focal Score* of terms have better effects on embeddings that carry document-level co-occurrence information. For the *Label Expansion* module, JE+SE outperforms Joint-Emb on both dimensions. IR+QE only outperforms IR on the topic dimension. It is because when expanding queries in IR, the country names have very close embeddings from word2vec. The expanded keywords for a country name is then other country names. Thus the classifier loses the ability to distinguish countries. Finally, when combining *Document Focalization* and *Label Expansion* modules, our method outperforms the ablations with only one module.

Table 4.2: Qualitative result of document allocation algorithms

Algorithm	*Topic Dimension*		*Location Dimension*	
	Micro-F1	*Macro-F1*	*Micro-F1*	*Macro-F1*
IR	0.3963	0.4520	0.3201	0.4173
IR+QE	0.4112	0.4744	0.2592	0.2894
Word2vec	0.5928	0.3890	0.2207	0.2339
Word2vec+DF	0.6101	0.3980	0.2378	0.2582
Topic Model	0.6264	0.3620	0.2120	0.1860
TMwP	0.7017	0.5120	0.3230	0.3230
Dataless	0.5882	0.3724	0.2510	0.1957
Joint-Emb	0.6938	0.4992	0.3383	0.3385
JE+DF	0.7862	0.5235	0.4269	0.4437
JE+LE	0.7347	0.5081	0.3472	0.3528
Ours	**0.7957**	**0.5413**	**0.4435**	**0.4520**

Table 4.3: Case Study on expanded terms for dimension labels

Dimension	Label	1st Expansion	2nd Expansion	3rd Expansion
Topic	*Movies*	films	director	hollywood
	Baseball	inning	hits	pitch
	Tennis	wimbledon	french open	grand slam
	Business	company	chief executive	industry
	Law Enforcement	litigation	law	county courthouse
Location	*Brazil*	brazilian	sao paulo	confederations cup
	Australia	sydney	australian	melbourne
	Spain	madrid	barcelona	la liga
	China	chinese	shanghai	beijing

Effect of Parameters

The key parameters in Dimension-Aware Joint Embedding are 1) the iteration number K_2 in *Dociment Focalization* and 2) the threshold η in *Label Expansion*.

Fig 4.3a (on topic dimension) and Fig 4.3b (on location dimension) show the performance change as K_2 changes during *Document Focalization*. We plot the two lines as 1) the Micro-F_1 change in JE+DF without label expansion and 2) the Micro-F_1 change in our approach with label expansion. In these two figures, we observe that the performance of *Document Focalization* improves most in the first two iterations and the change stablized after three iterations. The first iteration computes the *dimension-focal scores* using the doc-label similarity matrix $\mathbf{R}^{(\mathcal{DL})}$ that is derived from initial embeddings. Thus a lot of non-focal terms are down voted in iteration 1. Therefore, the improvement from the first iteration is the largest. The change of iteration 2 is also remarkable. That is because the doc-label similarity matrix $\mathbf{R}^{(\mathcal{DL})}$ after iteration 1 is improved due to the introduction of *dimension-focal scores*, the new round of *dimension-focal scores* using the updated doc-label similarity matrix is improved accordingly.

We are also interested in evaluating the label expansion threshold η, which controls the stopping criteria of label expansion. A bigger η indicates that less terms are added to represent the label. Fig 4.3c shows the change of Micro-F_1 as η changes using JE+LE. We observe that on both topic dimension and location dimension, the expansion brings improvement in the beginning, but worsen the performance as "too many" terms are expanded. It is reasonable because the initially added terms can enrich the semantics of the label and enlarge its semantics coverage. However, beyond a certain point, the more terms added, the more noise it brings, and the worse the performance gets. For example, one can add non-focal words like *money* or non-popular words like *Ben Bernanke* into label *Economy*, which leads to worse performance.

4.5.3 Case Study

Focal Score Analysis

In this section, we examine the quality of the focal scores in the 2-dimensional NYT dataset. In Table 4.4, we pick 6 terms as our example to justify the *Document Focalization* module. The first two terms, *economic growth* and *soccer*, both have very high focal scores on topic dimension but very low focal scores on location dimension. Thus these terms are emphasize when generating topic-aware representations and de-emphasized when generating location-aware representations. The terms, *chinese consumer* and *australian open champion*, have high focal scores on both topic dimension and location dimension. That means such terms have high discriminative power in both dimensions. Similarly, there are terms only discriminative for the location dimension, such as *beijing* and *new york state*. These terms do not carry any topic-related semantic but are very strong signals to decide the location allocation.

Table 4.4: Example of focal scores on different dimension

	Dimension-Focal Score	
Term	**Topic Dim.**	**Location Dim.**
economic growth	0.972	0.223
soccer	0.883	0.096
chinese consumers	0.999	0.994
australian open champion	0.999	0.698
beijing	0.245	0.681
new york state	0.166	0.788

Label Expansion Examples

We show in Table 4.3 the top expanded terms for 5 labels in topic dimension and 4 labels in location dimension. The expanded terms clearly show why the expansion is beneficial. For example, many documents describing *China* may not explicitly use the term *china*. In such cases, the expansion of terms like *chinese*, *beijing* and *shanghai* will enrich the semantic coverage of the label *China*, and thus improve cell-based document allocation.

4.6 Summary

We studied the novel problem of multi-dimensional cell-based document allocation. We proposed a label-efficient framework that requires only the surface names of different labels, and learns dimension-aware joint embeddings for achieving high accuracy in the allocation process. The experiments validate the effectiveness of the proposed method and its advantages for multi-dimensional document allocation over previous methods.

The proposed method, *i.e.*, Document Focalization, is a general solution to problems where additional label information is present. The separation of document semantic by different aspects (*i.e.*, dimensions) can benefit all kinds of unsupervised document representation learning methods, thus be a general foundation for many downstream text mining or NLP tasks.

Chapter 5

Comparative Analysis via Representative Phrase Mining

5.1 Overview

Prior work. Different from traditional OLAP in data warehousing or text cubes, CASeOLAP in text cubes requires contrasting one cell with other cells in order to find top-k representative phrases as the "measure" and thus poses many new challenges. We first examine some related work.

- [54, 4] studied phrase ranking for an arbitrary subset of documents and proposed Multidimensional Content eXploration (MCX). But their ranking was solely based on phrase frequency ratio in the subset and in the whole collection, ignoring the information in neighboring cells and thus cannot satisfy criterion (iii). Moreover, it uses frequency cutoff to find phrases and thus cannot satisfy criterion (i).
- [38] studied quality phrase mining from a large corpus and proposed a SegPhrase approach. Phrases are mined globally using the whole corpus statistics, and text segmentation was used to ensure criterion (i). However, the ranking was not tailored to a specific cell. With simple modification it can address criterion (ii), but not (iii).

Several insights are spotted when dissecting the problem. First, effective global phrase mining with the help of segmentation should be used for generating quality phrase candidates, instead of simple frequent word sequence mining. Second, although a comparison between local frequency and global frequency is useful for filtering much irrelevant phrases, it is inadequate to discover the most representative phrases. More comprehensive structural information of the multi-dimensional text cube (*e.g.*, neighborhood of target cell) should be preserved as context.

In this work, we systematically build our solution based on these insights. First, SegPhrase is employed to generate candidate phrases for the entire corpus. Second, a ranking measure is designed to respect all the three criteria. Specifically, the measure incorporates phrase distribution information from sibling cells (*e.g.*, $\langle$Japan, Economy$\rangle$) as context. While the ideas are intuitive, there are a number of challenging issues.

1. It is challenging to properly measure *distinctiveness* with sibling information due to (i) the large number of siblings, and (ii) the rather different number of documents that each cell may have. Our solution contrasts with MCX which only involves two frequency calculations, one at local (cell) level and one at global (full

collection) level. Moreover, the distribution for a particular phrase over sibling cells can often be sparse: our measure must be robust under sparse distribution in a vector space.

2. The second challenge is the computational constraint of a CASeOLAP operation. Computing all measures online can result in long latency since there are often tens of thousands of documents and phrase candidates. Furthermore, different from traditional OLAP aggregations, our context-aware aggregation requires the computation for sibling cells, which imposes extra complexity. New optimization technique for pre-computation (*i.e.*, *materialization*) is required.

This study contributes to text OLAP as follows.

- We propose a solution to support context-aware semantic OLAP in multi-dimensional text cubes. It is the first work that mines top-k representative phrases that are integral, popular and distinctive for a text cube cell posed by an ad-hoc query.
- We design a distinctiveness measure that leverages phrase distributions across neighboring cells of a target cell, and that produces fine-grain assessment. This measure is a key differentiator from previous phrase mining work. We creatively connect the problem of finding most distinctive phrases in a cell to a dual problem of finding the most relevant cell for each phrase. We are then able to address the above design challenge by leveraging both information retrieval and multi-class classification techniques.
- We develop both online and offline computational optimization. We use *early termination and skipping* to generate top-k phrases online efficiently. For offline materialization, we employ a *hybrid materialization* strategy: fully materializing lightweight phrase-independent statistics for all cells and partially materializing expensive phrase dependent statistics for selected cells. Due to the new challenge of coupled processing cost of sibling cells, we propose new heuristics that choose materialization order according to the *utility* in overall cost reduction. The technique can be generally applied to measures that require neighborhood cell statistics.
- Experimental results demonstrate that the proposed solution is effective and efficient. Top ranked phrases are representative and validated in both quantitative and qualitative evaluation. Materialization cost was reduced by 80% while all queries were answered within constrained time. Case studies suggest the usage of mined phrases in applications like text summarization.

The CASeOLAP framework is also applied to multiple real world scenarios and promotes productivities for multiple groups. Two cases will be discussed in Chapter 5. One is our collaboration with UCLA on *Clinical Biomarker Analysis* , the other is our collaboration with RPI on *Protest News Analysis*.

Table 5.1: Frequency statistics useful for composing popularity or distinctiveness measure

Definition	Meaning
$\sum_{d \in \mathcal{D}_c} tf(p,d) = tf(p,c)$	the frequency of phrase p in cell c
$\lvert\{d \lvert p \in d, \forall d \in \mathcal{D}_c\}\rvert = df(p,c)$	the count of documents in c that contain p
$\sum_{p \in \mathcal{D}_c} tf(p,c) = cntP(c)$	the total count of all the phrases that occur in c
$\lvert\mathbb{S}(c)\rvert = cntSib(c)$	the total count of sibling cells c has
$\max_{p \in \mathcal{D}_c} df(p,c) = maxDF(c)$	the maximum document frequency of any phrase in c
$\frac{cntP(c) + \sum_{c' \in \mathbb{S}(c)} cntP(c')}{cntSib(c)+1} = avgCP(c)$	the average counts of all phrases in cell c and its siblings

5.2 Preliminaries

First, we acknowledge that these three criteria can all be subjective and relative, and it is difficult to find a clear binary judgment whether each phrase satisfies all the criteria. Therefore, we decide to use a score between 0 and 1 to characterize the degree of each phrase in satisfying these criteria. For phrase p in cell c, we use $int(p,c) \in [0,1]$, $pop(p,c) \in [0,1]$, and $disti(p,c) \in [0,1]$ to denote the three criteria, and $r(p,c)$ to denote the overall ranking score that combines these criteria.

To combine the above criteria, we first notice that they reflect conjunctive conditions that should be satisfied, and one cannot replace the other. For example, popular word sequences may have quite low distinctiveness and sometimes ill-formed surface (*i.e.*, low integrity). Rare phrases that only occur once can be well distinctive. Since every criterion is indispensable, any low score (*i.e.*, near 0) in $int(p,c)$, $pop(p,c)$ or $disti(p,c)$ should result in a low rank for phrase p. Therefore, we design $r(p,c)$ as the geometric mean of those three scores.

$$r(p,c) = \sqrt[3]{int(p,c) \cdot pop(p,c) \cdot disti(p,c)} \tag{5.1}$$

The three criteria are equally positioned, though one can assign different weights according to user's requirement in different applications. If one of the factors is close to 0, the geometric mean will be close to 0 as well. Alternatively, one can use harmonic mean to have the same property, but the score will then be strongly dominated by the weakest factor, which may be unfavorable because the role of the other two factors will be neglected.

When we design the concrete measures for each criterion, we are aware that the input documents can be any textual word sequences with arbitrary lengths, such as articles, titles, queries, tags, memos, messages and records. A good design of the measures should generalize well to a variety of text data. Therefore, we tend to use more statistical features and fewer linguistic features.

Now we discuss design principles that are more specific to the three criteria.

- Popularity and distinctiveness of a phrase are dependent of the target cell, while integrity is not. Hence,

$int(p, c)$ can be simplified as $int(p)$.

- Popularity and distinctiveness can be measured from frequency statistics of a phrase in each cell, while integrity cannot. To measure integrity, one needs to investigate each occurrence of the phrase and other phrases to determine whether that phrase is indeed an integral semantic unit. Table 5.1 lists potentially useful frequency statistics that can be composed to calculate popularity and distinctiveness.
- Popularity relies on statistics from documents only within the cell $\mathcal{D}_c$, while distinctiveness relies on documents both in and out of the cell. We define the documents involved for distinctiveness measure calculation as *contrastive document set*. As we have seen in the above example, using the whole collection as contrastive document set is not accurate enough. More precise distinctiveness measure requires appropriate choice of contrastive document set.

The following subsections present our concrete design under these principles.

5.3 Phrase Mining Approach

5.3.1 Popularity and Integrity

Popularity indicates how significant the presence of a phrase has in the target cell. Very rare phrases in a cell should be lower in the ranking. However, the increase of phrase frequency should have a *diminishing return*. For example, a phrase occurring once is significantly less popular than a phrase occurring 11 times, but occurring 100 times or 110 times do not make a big difference. Therefore, we use the following formula as the measure for popularity.

$$pop(p, c) = \frac{\log(tf(p, c) + 1)}{\log cntP(c)} \tag{5.2}$$

Integrity is more complicated. It cannot be easily quantified from the simple frequency statistics. Fortunately, the SegPhrase technique [38] addresses it by combining global phrase mining and phrasal segmentation. The following example illustrates the key idea of phrasal segmentation.

Example 5.1 *Consider the following occurrences of* 'support vector machine' *and* 'support vector'.

1. *More formally, a* [support vector machine] *constructs a hyperplane...*

2. *The* [support vector] *method is a new general method of* [function estimation]...

The segmentation assigns every word occurrence to only one phrase, therefore non-integral phrase 'vector machine' *has no frequency count and* 'support vector' *is only counted in the second instance. The new frequency count based on the segmentation result is called rectified frequency.*

In a nutshell, SegPhrase first generates frequent phrase candidates according to global popularity requirements, and then estimates phrase quality based on multiple features. Most features are based on corpus statistics, and the only linguistic features used are English stopwords. Next, it estimates rectified frequency via phrasal segmentation and feed the segmentation-based feature back to the phrase quality estimator. It iteratively repeats the quality estimation and segmentation step until the best performance is achieved. We use the final quality estimator from SegPhrase as the integrity measure $int(p)$.

One additional benefit from SegPhrase is the rectified frequency based on segmentation. It is a better statistic to use than raw frequency because non-integral phrases get discounted. We use the rectified frequency to calculate the statistics in Table 5.1.

5.3.2 Context-Based Distinctiveness

In this section, we exploit the rich context naturally structured in the multi-dimensional cube, and develop a refined measure for distinctiveness. For illustration, we use a given user-specified query ⟨China, Economy⟩ in news cube as a running example throughout this section.

The notion of distinctiveness naturally involves comparing a phrase's occurrences in the documents of one cell with a set of contrastive documents. The first step of designing the measure is to determine what contrastive documents should be compared to. As defined in Sec. 1.4, a cell's context contains three parts: parent set, children set and sibling set. In addition, the whole collection is a natural contrastive set as well. We first eliminate the choice of children set $\mathbb{C}(c)$, since they do not provide any document outside the target cell. This leaves us three choices of contrastive document set to use: 1) entire collection, 2) cells in parent set, and 3) cells in sibling set. We examine the following example to find the right choice.

Example 5.2 *The statistics of phrase 'japanese stocks' given query ⟨∗, China, Economy⟩ is shown below.*

Cell	$cntP(\cdot)$	$tf(p,\cdot)$	$tf(p,\cdot)/cntP(\cdot)$
⟨∗, ∗, ∗⟩	*2.27M*	*11*	*0.00048%*
⟨∗, ∗, *Economy*⟩	*218,923*	*10*	*0.0046%*
⟨∗, *China, Economy*⟩	*14,039*	*4*	*0.028%*
⟨∗, *Japan, Economy*⟩	*4,272*	*5*	*0.11%*

For phrase *japanese stocks*, if we simply compare its occurrence probability in ⟨China, Economy⟩ with ⟨Economy⟩ and ⟨∗⟩, respectively, we find the phrase possesses a much higher occurrence probability in ⟨China, Economy⟩ over its parent and the entire collection, respectively. However, from human judgment we know that 'japanese stocks' is only remotely relevant to Chinese economy. In fact, if we study the

⟨Japan, Economy⟩ cell, sibling of ⟨China, Economy⟩, the phrase's occurrence probability is much higher. As contrastive documents, the sibling context contains more valuable information than parents or the entire collection, towards defining stronger distinctiveness measure.

In order to make the best use of sibling context, we need to solve the challenge that the sibling set of one cell may contain a large number of cells, and the challenge that a phrase's occurrence across these cells may be sparse. Simply using the ratio of any statistic between two cells will not work. Instead, we provide a new perspective of measuring the distinctiveness, by connecting it to a dual problem: classification of each phrase into a most relevant cell.

A distinctive phrase p for cell c should distinguish documents in c from documents in any sibling cell c'. In other words, if we measure the ***relevance*** between phrase p and any cell in $\{c\} \bigcup \mathbb{S}(c)$, the relevance mass should mostly concentrate on c. That means, if we softly classify phrase P into one of the cells in $\{c\} \bigcup \mathbb{S}(c)$ by relevance, the likelihood of its class label being c should be high. By this reasoning, we actually convert the problem of finding distinctive phrases in a cell into a problem of softly classifying each phrase into a cell by relevance.

Assume, the relevance score (will be developed in Sec. 5.3.2 and 5.3.2) of phrase p to cell c and its siblings have been computed as $rel(p, c)$ and $rel(p, c'), c' \in \mathbb{S}(c)$, where $rel(\cdot, \cdot) \geq 0$. Then we can adopt the well accepted *softmax regression* to compute a soft classification probability distribution for label set $\{c\} \bigcup \mathbb{S}(c)$. This function transforms any real value relevance score vector into a probability distribution. We can use the component for cell c, *i.e.*, the probability of p classified into c, as the distinctiveness measure. A straightforward application of softmax is as follows.

$$disti(p, c) = \frac{e^{rel(p,c)}}{\sum\limits_{c' \in \mathbb{S} \bigcup \{c\}} e^{rel(p,c')}} \tag{5.3}$$

According to this formula, the siblings where phrase p never occurs have a non-zero probability. It is counter-intuitive because the phrase should not be classified to the cells that it never appears in. Thus, we eliminate those cells from candidate class label set. This is equivalent to setting $rel(p, \cdot) = -\infty$ for those cells.

This modification triggers another problem. We are not able to distinguish two unique phrases that only appear in c with different relevance score. To solve this problem, we add a *none* label in the classification with relevance score always set to 0. The updated distinctiveness measure is:

$$disti(p, c) = \frac{e^{rel(p,c)}}{1 + \sum\limits_{c' \in \mathbb{S} \bigcup \{c\}, p \in c'} e^{rel(p,c')}} \tag{5.4}$$

This measure uses softmax to handle the challenge that we have multiple sibling cells as contrastive document sets, and modifies it to make it robust under sparse occurrence across siblings. It leaves flexibility in designing reasonable relevance score between a phrase p and a cell c. Intuitively, the relevance score should be positively correlated with the frequency $tf(p, c)$. Yet there are other factors one needs to consider to avoid undesirable biases. We present two design considerations in the following.

Balance Cell Size and Phrase Frequency

It is obvious that phrase frequency tends to grow with cell size, so it is intuitive to normalize the frequency by cell size. Unfortunately, we find that neither $tf(p, \cdot)$ nor occurrence probability $tf(p, \cdot)/cntP(\cdot)$ (*i.e.*, normalized frequency) is good enough. $tf(p, \cdot)$ favors large cells since they tend to have more occurrences of everything, and occurrence probability favors small cells.

Example 5.3 *For ⟨China, Economy⟩, 'double digit growth' is a distinctive phrase. However, the statistics of this phrase given query ⟨∗, China, Economy⟩ is shown as follows.*

Cell	$cntP(\cdot)$	$tf(p, \cdot)$	$tf(p, \cdot)/cntP(\cdot)$
⟨∗, China, Economy⟩	*14,039*	*8*	*0.057%*
⟨∗, US, Economy⟩	*135,947*	*12*	*0.0088%*
⟨∗, Emirates, Economy⟩	*142*	*1*	*0.70%*

Clearly, its frequency in ⟨China, Economy⟩ is lower compared to ⟨US, Economy⟩. Meanwhile, ⟨Emirates, Economy⟩ beats ⟨China, Economy⟩ by occurrence probability even though it only has one occurrence.

Our solution is to model each cell as a *super document* by concatenating its documents together, and resort to information retrieval techniques that fairly evaluate relevance for various-length documents. This problem has been extensively studied and the classic solution is BM25 [50]. Since we evaluate relevance all using the same phrase p, the IDF term in BM25 can be dropped. We define *Normalized TF*:

$$ntf(p, c) = \frac{tf(p, c) \cdot (k_1 + 1)}{tf(p, c) + k_1 \cdot (1 - b + b \cdot \frac{cntP(c)}{avgCP(c)})} \tag{5.5}$$

where k_1 and b are free parameters. We pick the commonly used configuration $k_1 = 1.2$ and $b = 0.75$ [40] to properly balance the cell size and phrase frequency. Here ntf is bounded by $k_1 + 1$. It goes up slower with tf when the cell is larger than average, and vice versa. It counters the bias of using only frequency or occurrence probability.

Document Frequency Matters

The above super document model for a cell flattens the internal structure of a cell. It cannot distinguish a phrase occurring many times in a single document from a phrase occurring in many documents with the same total frequency. Intuitively, the latter indicates a more relevant phrase because it covers more documents in the cell.

Example 5.4 *The following table shows the statistics of two phrases given query* $\langle *,$ *China, Economy*$\rangle$*. Even though phrase 'chinese consumers' and 'iron man' both appear 7 times in the cell, 'chinese consumers' is more spread-out in the cell and 'iron man' only appears in 2 documents.*

Cell	*Phrase*	$tf(p,\cdot)$	$df(p,\cdot)$
$\langle *,$ *China, Economy*$\rangle$	*chinese consumers*	7	6
$\langle *,$ *China, Economy*$\rangle$	*iron man*	7	2

Therefor, if p has large occurrences but only appears in a few documents, the relevance to the cell needs to be penalized, and vice versa.

Thus we propose the *Normalized DF* penalty:

$$ndf(p,c) = \frac{\log(1+df(p,c))}{\log(1+maxDF(c))} \tag{5.6}$$

The logarithm is used to place a nonlinear penalty. The phrases with very low document frequency are penalized more, and the phrases with very high document frequency are not overly rewarded. The denominator normalizes it into $[0,1]$. With this penalty factor, we define the relevance score of phrase p to cell c as:

$$rel(p,c) = ndf(p,c) \cdot ntf(p,c) \tag{5.7}$$

5.4 Optimization

Cost Estimation

In this section, the cost of collecting statistics is measured roughly by the estimated number of CPU clock cycles using the optimal strategy. The latency constraint $\mathcal{T}$ has the same unit and is used to compare with the estimated cost.

Since computing tf and df for all phrases in c have the shared counting process ($|\mathcal{D}_c|$-way merge join from $|\mathcal{D}_c|$ documents in a cell) and similar aggregation formula, they can be materialized with the same

manner and cost. Thus we can write the total cost of collecting statistics for queried cell c as:

$$\mathcal{Q}(c) = 2 \sum_{c' \in \mathbb{S}(c) \bigcup \{c\}} \mathcal{Q}_{tf}(c') \tag{5.8}$$

where $\mathcal{Q}_{tf}(c')$ is the cost of computing $tf(\cdot, c')$ for cell c'. $\mathcal{Q}_{tf}(\cdot)$ of siblings are included here as sibling statistics are also required for computing representative phrases in cell c.

We show how $\mathcal{Q}_{tf}(c)$ can be recursively estimated in the cell space, for a given cell $c = (a_1, \ldots, a_n, \mathcal{D}_c)$, where $a_i \in \mathcal{A}_i$ (including '$*$'). Without loss of generosity, we assume $des(a_i) \neq \emptyset$ for $1 \leq i \leq n' \leq n$. Thus we have n' aggregation choices; *i.e.*, aggregating cells in one of the following subcell set:

$$S(c)_i = \{c_i = (a_1, \ldots, a, \ldots, a_n, \mathcal{D}_{c_i}) | a \in des(a_i) \wedge \mathcal{D}_{c_i} \neq \emptyset\}$$

Each subcell set $S(c)_i$ of c contains subcells by replacing i-th dimension value to its descendants. Other than aggregating from subcells, one choice is to gather $tf(p, c)$ from raw text. The optimal choice should be selected for online computation if the cell is not materialized. Hence, the optimal cost among the $(n' + 1)$ choices should be used for our estimation.

As shown by previous work [37], the OLAP query within the cell space has the *optimal substructure* property. As a consequence, *dynamic programming* can be used for computing optimal cost and choice of aggregation:

$$\mathcal{Q}_{tf}(c) = \min \left\{ \mathcal{Q}_{raw}(c), \min_{1 \leq i \leq n'} \left\{ \mathcal{Q}_{agg}(S(c)_i) + \sum_{c' \in S(c)_i} \mathcal{Q}_{tf}(c') \right\} \right\}$$

where $\mathcal{Q}_{raw}(c)$ and $\mathcal{Q}_{agg}(S)$ denote the cost for merging counts from raw text and aggregating from subcell set $S(c)_i$ respectively. Let λ_c denote the average number of unique phrases in each document in c, we calculate them as follows.

$$\mathcal{Q}_{raw}(c) = \lambda_c |\mathcal{D}_c| \log |\mathcal{D}_c| \tag{5.9}$$

$$\mathcal{Q}_{agg}(S(c)_i) = \sum_{c' \in S(c)_i} |\mathcal{P}_{c'}| \tag{5.10}$$

Eq. (5.9) is obtained by performing a $|\mathcal{D}_c|$-way merge join [4] in the documents contained in cell c. In particular, it scans the sorted phrase lists of documents in parallel. During the merge, $df(p, c)$ can also be counted by the number of lists where p is seen. Equation (5.10) is derived by merging phrase statistics from the subcells in S to the target cell c. Hashmaps are used to guarantee the lookup cost and insertion cost are $\mathcal{O}(1)$.

For a precomputed cell or empty cell, we define:

$$\mathcal{Q}_{tf}(c) = 0 \quad (c \text{ is materialized or empty}) \tag{5.11}$$

There is one most prominent difference of our query processing cost structure compared with previous OLAP work. The cost for computing context-aware distinctiveness score for any query is tied to the cost of computation for context cells (siblings in our case), rather than just the target cell. This can be seen from Eq. (5.8). It is a general property for any *context-aware measure in OLAP*. This new property poses an interesting new challenge to traditional greedy materialization strategy, as the computational cost of sibling cells become *coupled*. We first present an algorithm that ignores this challenge, and then propose a better algorithm to address it.

5.4.1 Hybrid Offline Materialization

We propose a more refined materialization plan, which does not materialize all siblings at once when a cell fails to meet $\mathcal{T}$. Instead, it repeatedly attempts materialization of one sibling, and reevaluates the cost of querying the target cell, until it falls below $\mathcal{T}$. The order of choosing siblings affects how many siblings will be materialized and how much storage cost is needed to meet the constraint. We use a *utility* function for each sibling cell c' to guide this process. Intuitively, we have the following choices of utility function.

1. cost reduction to the target query $\mathcal{Q}_{tf}(c')$;
2. cost reduction to all queries $\mathcal{Q}_{tf}(c')(|\mathbb{S}(c')| + 1)$;
3. cost reduction to critical queries which haven't met the constraint $\mathcal{Q}_{tf}(c')|\{c \in \mathbb{S}(c'), \mathcal{Q}(c) \geq \mathcal{T}\}|$; and
4. cost reduction to all queries per storage unit $|\mathbb{S}(c')|$;
5. cost reduction to critical queries per storage unit $|\{c \in \mathbb{S}(c'), \mathcal{Q}(c) \geq \mathcal{T}\}|$.

The cost reduction to the target query per storage unit is a constant 1, which cannot provide any guidance.

The choices 2–5 all reflect the cost reduction beyond the target query. Due to the neighborhood coupling, the computational benefit of a particular cell is shared by neighboring cells, *i.e.*, siblings in our task. Since the sibling relationship is mutual (c's siblings must have c as sibling as well), the pre-computation of c' reduces the cost querying siblings of c', and querying itself. Hence we have the factor $(|\mathbb{S}(c')| + 1)$ in choice 2. Choice 3 is similar, except that it values the cost reduction only to the queries that currently cannot be answered within time $\mathcal{T}$. Choice 4 and 5 normalize the cost reduction by the storage cost of materialization,

which measures the unit gain. This refined version may requires to monitor $\mathcal{Q}(\cdot)$ of unexamined cells to compute the utility function. According to the definition of sibling, the siblings of cell c share the same cuboid of c. Therefore we cope with this by grouping non-empty cells into cuboids and estimate $\mathcal{Q}_{tf}(\cdot)$ and $\mathcal{Q}_1(\cdot)$ of all cells in the cuboids before materializing any of them. In the concrete algorithm, one of the five *utility* functions is used to provide different balance between query time and storage. The utility-guided algorithm also guarantees the latency requirement.

The proposed utility guided algorithm is described as follows.

Algorithm 5.1: Utility-Guided Greedy Algorithm for Partial Materialization

```
1  utility(·): the chosen utility function
2  T: latency constraint for any online query
3  C: cuboid list in bottom-up topological order
4  for cuboid B in C do
5      for cell c in cuboid B do
6          estimate Q_tf(c) and Q_1(c) based on Eq. (5.8)-(5.11)
7      for cell c in cuboid B do
8          Q_1(c) = 0
9          for c' ∈ S(c) ⋃ {c} do
10             if c' is not materialized then
11                 Q_1(c) += Q_tf(c')
12         Q_1(c) = Q_1(c) * 2
13         if Q_1(c) ¿ T then
14             initialize U ;                          // sorted array of utilities
15             for c' ∈ S(c) ⋃ {c} do
16                 add (c', utility(c')) into U
17             while Q_1(c) ¿ T do
18                 c' = U.pop() ;                      // pop the maximum cell
19                 Q_1(c) -= 2 * Q_tf(c')
20                 materialize c' using best strategy
21                 Q_tf(c') = 0
```

5.4.2 Optimized Online Processing

In online computation, we propose an early termination and skipping technique to prune phrase candidates that are impossible to be among top-k. Observe that distinctiveness score is the only more expensive measure since it involves computation of siblings. This inspires us to decompose the overall ranking measure into two parts: $disti(p, c)$ and $pop(p, c) \cdot int(p)$, where the latter part can be computed cheaply and serve as the upper bound of the ranking score. Moreover, if we estimate a more accurate upper bound of $disti(p, c)$, we can also derive a tighter bound for the overall ranking score, and largely prune the phrase list.

We first sort all phrase candidates by $u_1(p, c) = pop(p, c) \cdot int(p)$, and go through them one by one. That is, phrases with high cell popularity and integrity get evaluated early. As soon as the next phrase p has a

lower u_1 than the lowest final score θ of phrases in the top-k list, it is safe to terminate the enumeration. Otherwise, we estimate a tighter upper-bound $u_2(p, c)$ without using siblings' phrase-level statistics.

$$u_2(p, c) = e^{rel(p,c)}/(1 + e^{rel(p,c)}) \tag{5.12}$$

The computation of u_2 does not involve siblings of c. If $u_1(p, c) \cdot u_2(p, c) < \theta$, we can skip the actual computation of distinctiveness score and move on to the next candidate. In the worst case, we have to retrieve sibling statistics, which involves aggregations for non-materialized sibling cells.

5.5 Experimental Result

The dataset is constructed from 1976-2015 New York Times articles[1]. It contains 4,785,990 articles (2.28 billion tokens) covering various topics. The raw size of the textual content is 17.04 GB. These articles are already partially annotated by data provider. We use these annotations and *Named-Entity Recognition* to construct 6 dimensions: Topic, Location, Organization, Person, Year and DocType. The first four dimensions have more than one layer in the hierarchies and the last two dimensions are flat. In flat dimensions, all the values are regarded as siblings.

5.5.1 Effectiveness Evaluation

Phrase-To-Cell Allocation Accuracy

This task is designed to quantitatively assess how many top-k phrases are correct representative phrases. We test eight queries. Four of them are *1-Dim Queries*, and the other four are *2-Dim Queries*. To generate non-trivial test queries, we first randomly pick two *1-Dim Queries* and two *2-Dim Queries*; then for each picked query, we add the most similar sibling in terms of both size and content as a paired query.

To ease the labeling, for each pair of test queries, we first collect all top-50 phrases generated by all the measures for both queries. For each phrase in the pool, we label it with either one of the two cells which it best represents, or a 'None' label in three circumstances: 1) it is not a valid phrase, 2) it is not relevant to either cell and 3) it is a background phrase that are shared by both cells. We then measure the accuracy of phrase allocation by the average precision from top-5 to top-50 phrases. We show the result in Figure 5.1 for baselines and Figure 5.2 for ablations.

In general, as k grows, the precisions of those measures go down. In Figure 5.1, **RepPhrase** has

[1] http://developer.nytimes.com/

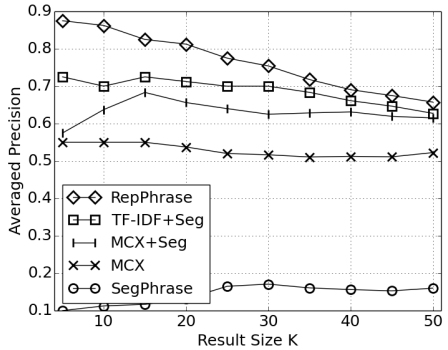


Figure 5.1: Phrase allocation accuracy comparison to baselines

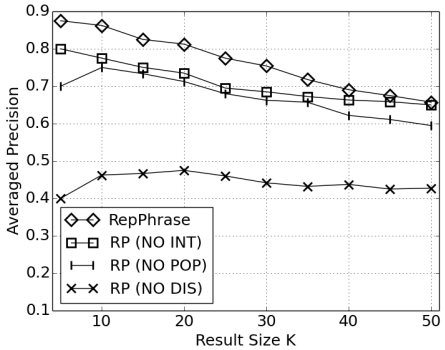


Figure 5.2: Phrase allocation accuracy comparison to ablations

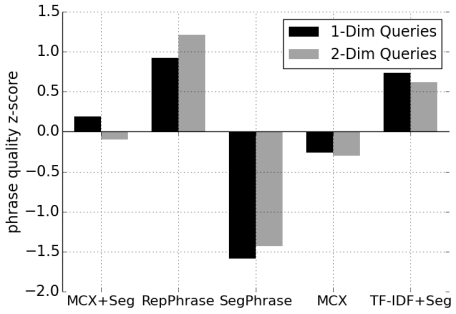


Figure 5.3: Phrase list quality comparison between baselines

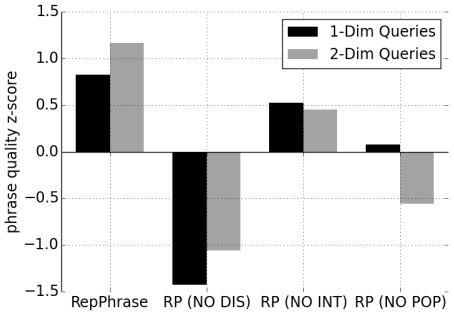
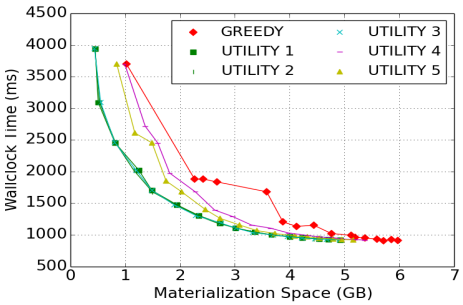


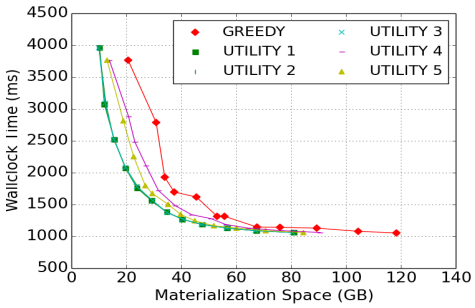
Figure 5.4: Phrase list quality comparison between ablations

the best precision and **SegPhrase** has the worst. Also, the difference of precision between **RepPhrase** and others decreases as k grows. That is attributed to the limited number of true representative phrases. **RepPhrase** successfully ranks these good phrases high, others gradually include them as k grows. Amongst all the baselines, **TF-IDF+Seg** outperforms others since it is the only baseline that captures all three criteria. However, it still loses to **RepPhrase**. Both use sibling cells as contrastive group, using classification probability (**RepPhrase**) as *distinctiveness* performs better than using IDF (**TF-IDF+Seg**).

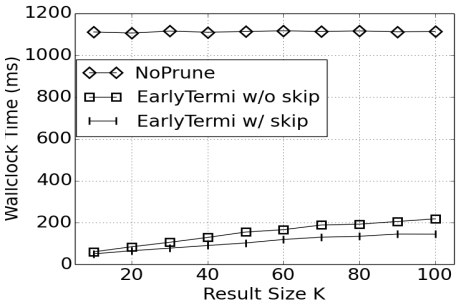
In Figure 5.2, we show the performance drop by removing one of the three criteria respectively. We notice that **RP (NO INT)** has the best precision amongst all ablations and **RP (NO DIS)** has the worst, which indicates the relative importance of the criteria: *distinctiveness* > *popularity* > *integrity*. One interesting comparison is between **MCX+Seg** and **RP (NO POP)**. These two can be viewed as two versions of standalone *distinctiveness* measure with different contrastive document groups. Using dynamic sibling cells as contrastive group (**RP (NO POP)**) performs better than using the static entire collectio n (**MCX+Seg**), especially on the top phrases. It further justifies the choice of using dynamic background over static background.



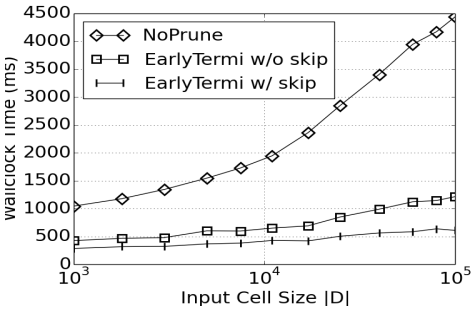
(a) Time-space balance of 4-Dim Cube



(b) Time-space balance of 6-Dim Cube



(c) Wallclock time (ms) for varying k



(d) Wallclock time (ms) for varying $|D|$

Figure 5.5: Performance of materialization optimization and online optimization

5.5.2 Efficiency Evaluation

We evaluate the computational performance using the full NYT dataset. For the offline computation, we compare the following algorithms for materializing phrase-level statistics: 1) **FULL** (full materialization), 2) **LEAF** (leaf materialization), 3) **GREEDY** and **UTILITY 1-5**.

For online computation, we compare three algorithms **NoPrune** (no pruning), **EarlyTermi w/o skip** and **EarlyTermi w/ skip**.

Materialization Evaluation

Two key metrics for a materialization approaches are (i) storage space, and (ii) worst query time. In order to study the materialization strategies in various scales, we create several databases by varying the number of dimensions it includes. Six text databases are constructed by gradually adding 6 dimensions, we name them **1,2,3,4,5,6-Dim Cube**, respectively. Even though they share the same raw textual data, the different dimension settings make the materialization process quite different.

Figure 5.5a and 5.5b show the space-time trade-off on **4-Dim Cube** and **6-Dim Cube**. Since **LEAF** and **FULL** strategies have quite exceptional worst query time or materialization space, the result is separately shown in Table 5.2. In **4-Dim Cube**, we first notice that the space cost of **LEAF** is as low as 0.68 GB,

Table 5.2: Space-time trade-off of LEAF and FULL

	4-Dim Cube		6-Dim Cube	
	Space (GB)	Time (s)	Space (GB)	Time (s)
LEAF	0.68	73.2	26.76	3407.5
FULL	20.17	0.86	706.0	0.89

but the worst query time is more than 73 seconds. If we materialize every cell as in **FULL**, it has the minimized worst query time but consumes about 20 GB to materialize. The other 6 strategies make trade-offs between time and space by setting different latency constraint $\mathcal{T}$. We notice that all five utility-guided strategies outperform **GREEDY**, *i.e.*, their curves are closer to the origin point. In particular, picking any of **UTILITY 1-3** yields the best trade-off that can take less than 10% of the storage compared to **FULL** and less than 50% of the **GREEDY** strategy with same worst query time. The reason that **UTILITY 1-3** performs better than **UTILITY 4-5** is that the first three utility functions reward the total cost reduction instead of cost reduction per record. It turns out that **UTILITY 1-3** tend to avoid materializing a huge sibling, and choose to materialize a smaller sibling with lower cost reduction per record, yet good enough to satisfy constraint $\mathcal{T}$. In practice, setting $\mathcal{T} = 6 \times 10^7$ achieves good space-time trade-off, where the worst query time is below 1.5 seconds and the materialization space is below 2G in **4-Dim Cube** and below 40G in **6-Dim Cube**.

5.5.3 Real-world Case: Clinical Biomarker Analysis

The CASeOLAP framework is also applied to multiple real world scenarios and promotes productivities for multiple groups. One is our collaboration with UCLA on Clinical Biomarker Analysis. Here is the multi-dimensional data cube summary.

Dataset: **CLINICAL BIOMARKER PAPERS**			
A collection of *PubMed* research papers about Cardiovascular Diseases from 1995-2016, with the top 250 highly relevant proteins to CVDs as target entities.			
# Documents	**# Dimensions**	**Avg. Word Count**	**Text Type**
500K	2	∼2000	Academic
Dimension	**# Values**	**Description**	
Disease	6	Six main Cardiovascular disease groups with MeSH terms	
Year	12	Different years from 1995 to 2016	

Table 5.3: Clinical Biomarker Papers

The objective of this collaboration is to apply CASeOLAP into Cardiovascular Disease analysis. Medical experts from UCLA are trying to spot the relationship between diseases and principle proteins. CASeOLAP compute the representative proteins for each disease cell and have the following analysis.

Disease	Top Ranked Protein (Gene Symbol)	CaseOLAP Score
Arrhythmia	Methionine synthase (MTR)	3.799
	Ryanodine receptor 2 (RYR2)	3.358
	Potassium voltage-gated channel subfamily H member 2 (KCNH2)	2.728
	Gap junction alpha-1 protein (GJA1)	1.873
	Platelet-activating factor acetylhydrolase (PLA2G7)	1.740
Congenital Heart Disease	Fibrillin-1 (FBN1)	4.925
	Plakophilin-2 (PKP2)	3.215
	Tyrosine-protein phosphatase non-receptor type 11 (PTPN11)	2.671
	Arachidonate 5-lipoxygenase-activating protein (ALOX5AP)	2.040
	Catechol O-methyltransferase (COMT)	1.789
Cardiomyopathy	Interferon gamma (IFNG)	3.341
	Interleukin-4 (IL4)	2.814
	Interleukin-17A (IL17A)	2.733
	Tumor necrosis factor (TNF)	2.550
	Titin (TTN)	2.353
Cerebrovascular Accident	Alpha-galactosidase A (GLA)	5.911
	Brain-derived neurotrophic factor (BDNF)	5.597
	Tissue-type plasminogen activator (PLAT)	4.948
	Apolipoprotein E (APOE)	3.690
	Methylenetetrahydrofolate reductase (MTHFR)	2.714
Ischemic Heart Disease	Cholesteryl ester transfer protein (CETP)	4.606
	Apolipoprotein A-I (APOA1)	3.994
	Adiponectin (ADIPOQ)	3.652
	Lipoprotein lipase (LPL)	3.304
	Myeloperoxidase (MPO)	3.242
Valve Disease	Mineralocorticoid receptor (NR3C2)	3.277
	Elastin (ELN)	2.383
	Tropomyosin alpha-1 chain (TPM1)	2.333
	Myosin-binding protein C, cardiac-type (MYBPC3)	1.702
	Platelet-activating factor acetylhydrolase (PLA2G7)	1.611

Figure 5.6: Top 5 Proteins in 6 main groups of Cardiovascular Disease According to Their CaseOLAP Scores

Principle Protein Lists

As shown in Figure 5.6, the top 5 proteins are displayed according to their CaseOLAP scores within each CVD group. Top ranked molecules in IHD were Cholesteryl Ester Transfer Protein (4.606), Apolipoprotein A-I (3.994) and Adiponectin (3.652), indicating a strong relevance of lipid metabolism, as well as neutrophil granulocytes (inflammatory response). In CVA, -galactosidase had a high score of (5.903), suggesting that glycoproteins and lipids are highly relevant. Interestingly, the majority of proteins in CM were inflammatory molecules, such as - Interferon (3.341), TNF (2.550), and Interleukins.

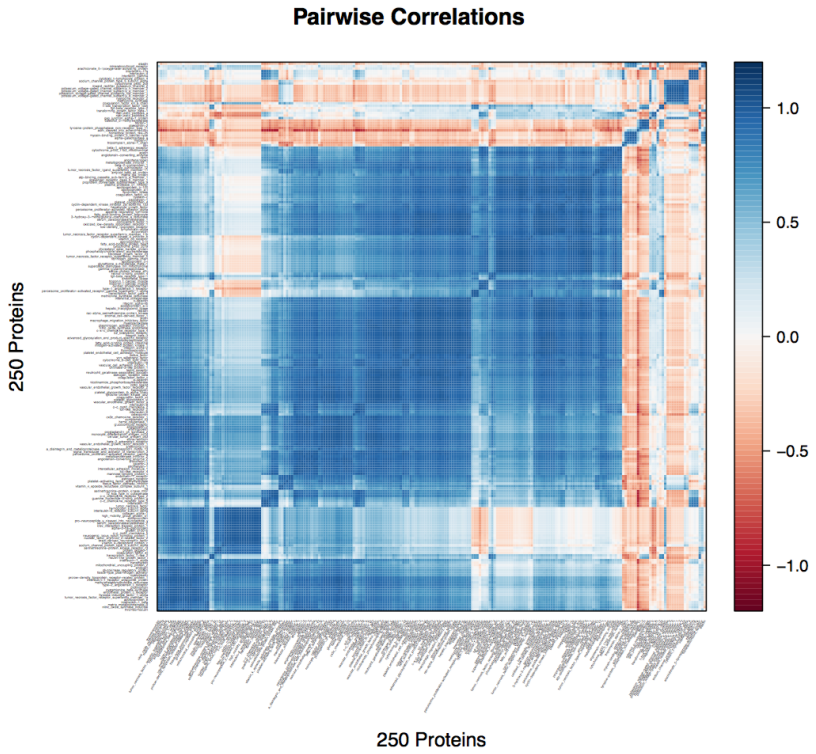


Figure 5.7: Heatmap of the top 250 Proteins in a Cross-Correlation Analysis

Proteins Cross-Correlation Analysis

As shown in Figure 5.7, using the CaseOLAP scores for each protein over 6 CVDs, we performed a cross-correlation analysis to identify 8 main protein clusters, which are colored blue. We found that each identified cluster represents key biological functions related to CVDs.

Top Protein Details

As shown in Figure 5.8, the top 25 proteins in CM exhibit a similar score pattern in both IHD and CVA with the majority of proteins revealing an inflammatory function. Contractile proteins, such as titin, have a high relevance in CM, as well as VD, but not in the other CVDs. Troponin-I had a very high score in IHD, as this is a popular biomarker in ischemic events, however, had little relevance in CVA. Intriguingly, amyloid beta A4 protein (an established protein in Alzheimer disease) and platelet-activating factor acetylhydrolase (a leukocyte function regulator), appear to be consistent over all six CVD groups.

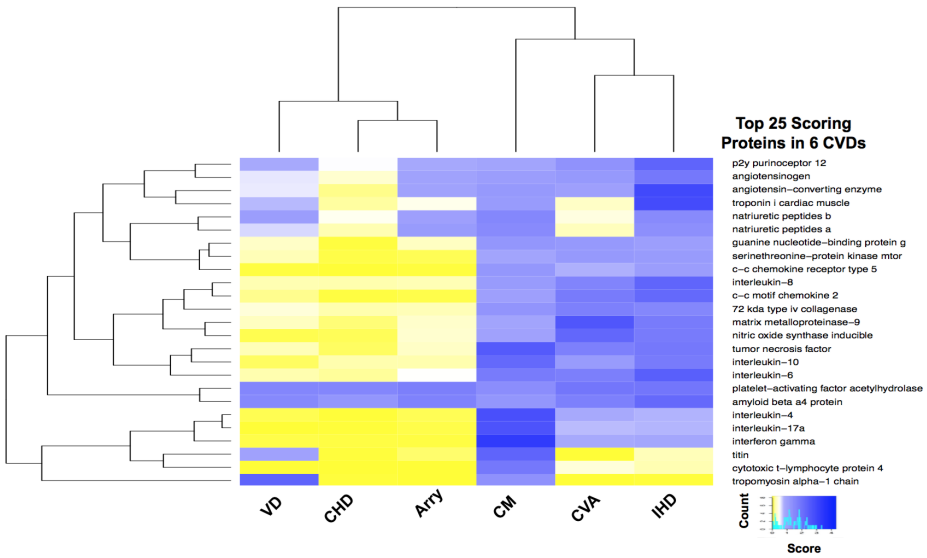


Figure 5.8: Top 25 Scoring Proteins in 6 groups of Cardiovascular Disease

5.5.4 Real-world Case: Protest News Analysis

Another is our collaboration with RPI on Protest News Analysis. Here is the multi-dimensional data cube summary.

Dataset: **PROTEST NEWS ARTICLES**			
A collection of Protest News Articles containing 59 protest incidents (January 2009 - December 2010) and 52 attack incidents (January 2014 - December 2015.)			
# Documents	**# Dimensions**	**Avg. Word Count**	**Text Type**
10K	6	∼500	News

Dimension	**# Values**	**Description**
Incident	111	Individual protest and attach incidents
Location	20	Countries that the protests happened
Type of Protest	6	Six different types of pretest such as *Demonstration*
Demands of Protest	4	Four different types of pretest such as *Political* and *Environmental*
Protester	10	Different protester groups such as *students* and *political opposition*
Time	48	Different months spanning from Jan 2009 to Dec 2015

Table 5.4: Protest News Articles

In this collaboration, we modeled the news articles related to protest events into a multi-dimensional text cube, where six dimensions are extracted: *Incident*, *Location*, *Type of Protest*, *Demands of Protest*, *Protester* and *Time*. We built an interface to support analysts' multi-dimensional query. For a given query, *e.g.*⟨Iran, Political⟩, we extract representative phrases and sentences for news articles about protests that happened in Iran and are of political demands.

As shown in Figure 5.9, the automated summary is generated by CASeOLAP where the sentences are extracted based on CASeOLAP score of each representative phrase. The phrases are also highlighted. The

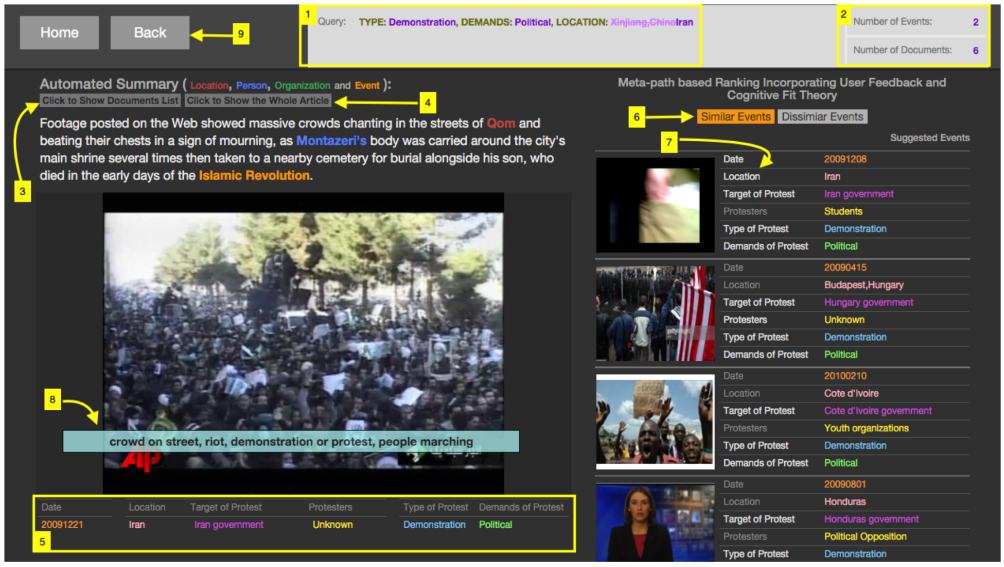


Figure 5.9: Protest Analysis Screen Shot with Representative Phrases / Sentences extracted for a specified cell.

right hand side has the similar cells discovered by measuring the overlap of representative phrases. The screenshot shows the example of query ⟨Iran, Political, Demonstration⟩.

5.6 Summary

In conclusion, this chapter proposes the first solution to support representative phrase mining in multi-dimensional text databases. It mines top-k representative phrases based on three criteria: integrity, popularity and distinctiveness. We also develop efficient online and offline computational optimization to empower phrase analysis in very large text databases.

The potential of this technique is far beyond phrase analysis. Our preliminary experiments also show that the resulting phrases can help compose high quality sentence-level summaries. Since the mined phrases characterize document subsets properly, they can be further exploited to facilitate the construction of multi-dimensional text databases. We plan to explore these research directions in the near future.

Chapter 6

System Design: Multi-dimensional Search and Mining

6.1 Overview

The EvenCube project has been funded by NASA, developed in the Department of Computer Science, UIUC, assisted by NASA and Boeing researchers, and later it has also been partially supported by ARL (Army Research Lab) via NSCTA (Network Science Collaborative Technology Alliance) program, with many other researchers joined in. With years of research and development, it has reached to a mature stage: Many interesting functions have been developed and integrated into the system and its power can be demonstrated on a spectrum of diverse datasets. The system was originally designed for analysis of ASRS (Aviation Safety Report System) data sets, nevertheless, information extraction tools have been used to extract structured entities from the typical news datasets, collected from multiple news agencies. Therefore, the system becomes a more general platform for construction, search, OLAP (online analytical processing) and data mining on integrated text and structured data.

The system provides multiple construction, search and mining functions with the following architecture.

System Architecture. The EventCube system is designed with the architecture shown in Figure 1. It consists of the following modules: (1) *Data Uploading and Preparation*, which pre-processes the free text corpus from user's uploading and converts it into a text-rich data cube with dimensions and topic hierarchy extracted; (2) *Indexing and Materialization*, which builds indexing and partial materialization results for phrase-based summarization, keyword search, top cell finding, single dimension distribution and hierarchical topic modeling; (3) *Query-Based Search and Mining Module*, which processes user-queries (both search and analysis queries) by parsing the query, selecting and executing appropriate search or mining module (which searches or mines on the constructed text-rich data cube to derive results); and (4) result presentation by *Visualization and Interpretation* of the search/mining processes and results. An screenshot of the system is shown in Figure 6.2

Substantial research have been conducted for the development of the EventCube system, with multiple technologies developed, including CASeOLAP [60], TextCube [37], TopicCube [68], TopCells [18], and TEXplorer [70], among others. These technologies have been incorporated into the system. Moreover, multiple

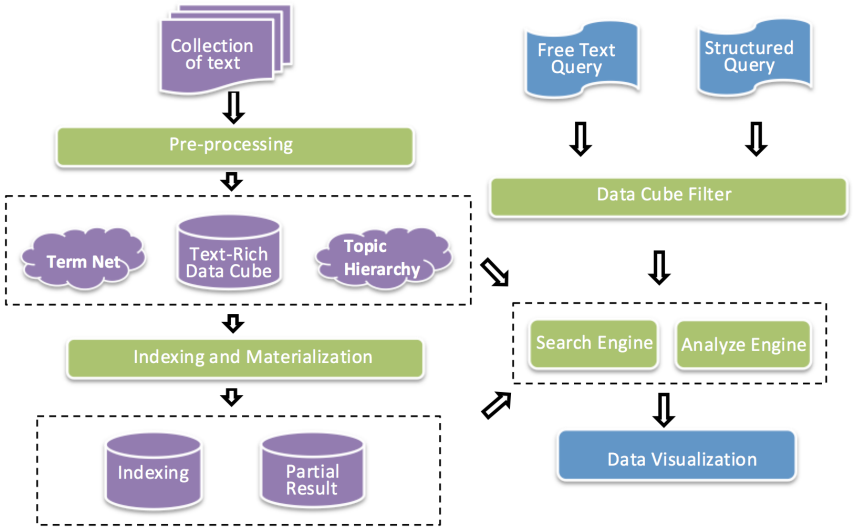


Figure 6.1: System Architecture of EventCube

powerful, efficient, and flexible construction, search, mining and optimization mechanisms have been incorporated into the system in a user-friendly manner. The system has high potential to be extended in many powerful ways and serve as a general platform for experiments of multidimensional search and mining of text and structured data.

6.2 EventCube Background

In this demo, we build a complete automatic workflow to construct and analyze a collection of text data. A user can easily browse different datasets to be analyzed in a dataset portfolio page. Also, she can upload their particular text corpus in an easy-to-use way. The system then creates an independent thread to extract entities/dimensions, construct the term-net, find the topic hierarchy, build indexes and compute partial materialization results as a background process. When the preprocessing is done, the system will inform the user and make the dataset accessible. A user is then permitted to conduct search, summarization and analysis tasks.

6.3 Major Functional Modules

6.3.1 Text Cube Construction

EventCube supports fully automated cube construction process with limited user guidance. In a nutshell, the cube construction module takes a collection of documents as input, discover both entity-based dimensions,

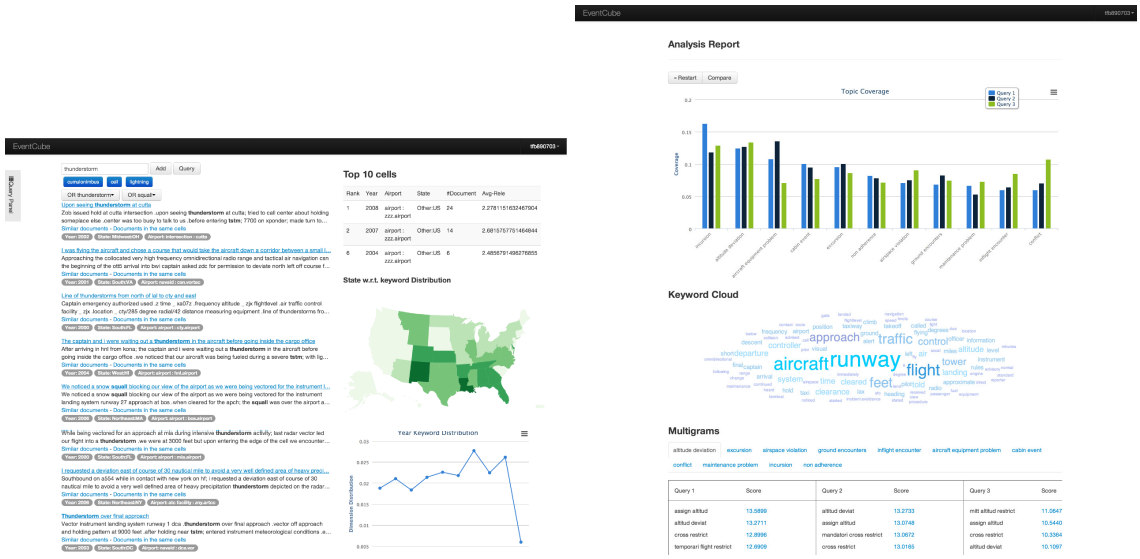


Figure 6.2: Screenshot of EventCube applied on Aviation Safety Report data

such as Location and Person, and hierarchical topic dimension. The whole construction keeps human in the loop that experts can interactively modify the extracted dimensions to meet their best interests. An example of three extracted dimensions on a news article corpus is presented in Figure 6.3.

Dimension-based Structure Creation

The *dimension-based structure creation* discovers dimensions. In EventCube, we supports two ways of finding meaningful dimensions.

- **Entity-based Dimension:** As discussed in Chap 3, we provide a couple of seeds for an entity-based dimension. The system automatically find all other entities that share similar nature of the proposed seeds using lexical pattern signals. The experts have flexibility to add or remove the resulting entity list. An example is shown in Figure 6.4 left, where *Location* dimension is constructed by providing two seeds: China and USA. Then 14 countries are extracted from news articles.

- **Topic Taxonomy Dimension [67]:** Another type of dimension does not rely on a particular type of entities mentioned in the text. The topics, on the other hand, are more subtle subjects that requires better semantic understanding to discover. We support the topic taxonomy discovery to iteratively generate a hierarchical topic dimension. Experts are required to provide their desired hierarchy height and value count. The module read all documents and construct a topic hierarchy. An example of a 3-layer topic hierarchy is presented in Figure 6.4 right. The system allows user to change the surface name of each discovered topic node.

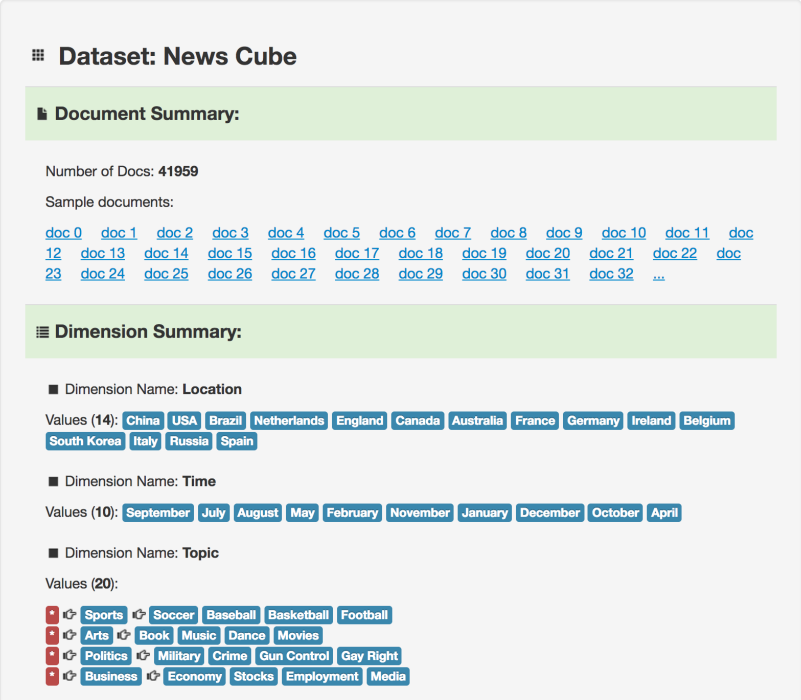


Figure 6.3: Example of constructing a NYT text cube with three dimension automatically created: Location, Time and Topic.

Cell-based Document Allocation

We apply the technique presented in Chapter 4 to allocate the documents into the discovered multi-dimensional cube space. Each document is assigned to a dimension value (can be '$*$' if none matches) for every dimension. Noting that the document allocation is supported to be incrementally executed. As new documents streaming in, the allocation can be applied to the newly added documents and enrich the extracted text cube.

6.3.2 Structure-Rich Search

Contextual Search

Compared to traditional search engine, *i.e.*, bag of words model, the search engine in EventCube supports a more intelligent contextual search function. Unlike general search such as Google Search, each search query in our system will only focus on a particular dataset with some particular domain knowledge. Therefore, the system builds a term network which includes the frequent mentioned entities, events and phrases in the corpus. Based on the term-net, we (1) recommend related terms to user's input terms to help improving their queries, (2) support AND, OR, NOT three boolean operators to compose advanced query, and (3)

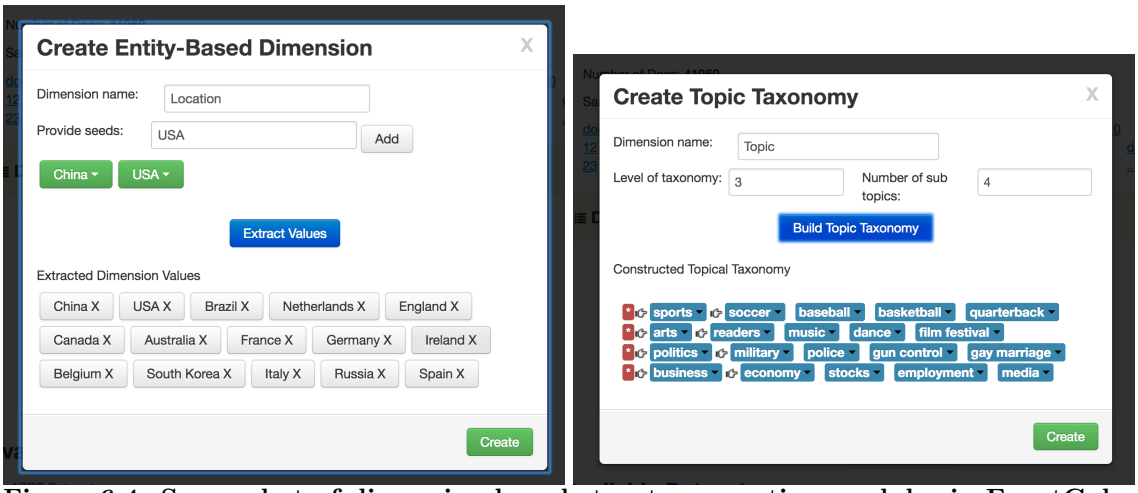


Figure 6.4: Screenshot of dimension-based structure creation modules in EventCube

include the equivalent terms, *e.g.*, abbreviations, as a part of query by default.

Besides query for short terms, EventCube also supports bulky text search. Inputting a long text, the system will extract frequent terms from the text, locate them in the term network, improve the query by adding equivalent terms and highly related terms. Based on this principle, the 'Similar Docs Search' function in EventCube can have the ability to find semantically similar docs, even when they contain totally different words distribution.

Top-Cell Finding

A cell in the text cube aggregates a set of documents with matching dimension values on a subset of dimensions. Given a keyword query, our goal is to find the top-k most relevant cells in the text cube. A relevance scoring model and efficient ranking algorithm has been proposed in [18]. It optimizes the search order and prunes the search space by estimating the upper bounds of relevance scores in the corresponding subspaces, so as to explore as few cells as possible for finding top-k answers. An example on the News dataset to generate top-k cells is shown in Figure 6.5.

Single Dimension Distributions based on Keywords

For each search query, it is desirable to provide many insights for analyzers if the data distribution can be provided on each dimension. In EventCube, we aggregate the relevant documents on every dimension to show the heatmap of the keywords for geographic dimension, time series of the keywords for time dimension and the ranked list for other dimensions. An example on aviation safety data can be found in Figure 6.6.

To achieve both efficiency and effectiveness, we propose a framework that combines offline and online computation together to generate real-time single dimension distribution results for every query.

Rank	Year	Event	Organization	#Document	Avg-Rele
1	201001	Movement:Transport	White House	8	12.995336890220642
2	201001	Life:Die	White House	6	11.930548350016275
3	201001	Movement:Transport	Navy	4	13.26450800895691
4	201001	Personnel:Elect	Congress	3	7.3021542231241865
5	201001	Contact:Meet	White House	3	13.659941037495932
6	201001	Personnel:Start-Position	United Nations	3	13.533504803975424

Figure 6.5: Top Cell List for keyword *haiti earthquake* on dimensions of *Year*, *Event*, *Org* in the News data

In the offline part, we first map equivalent terms into one, then build both keyword-doc inverted index and cell-doc inverted index, finally merge them to calculate single term distribution for each pair of terms and dimensions. In the online part, we match each term in user's query into its equivalent term. Based on the assumption of independency of terms, we use De Morgan's laws to estimate the combined term-dimension distribution.

6.3.3 Topical Analysis

Probabilistic topic models are among the most effective approaches to latent topic analysis and mining on text data. On the other hand, online analytical processing (OLAP) techniques are useful for analyzing and mining structured data, such as data cubes. By combining OLAP and topic modeling together, we treat hierarchical topic distribution as one of the aggregation functions. With the support of comparison of topic distribution for different cells, our system can provide deeper insights for decision makers. This is realized efficiently by our TopicCube model [68] that constructs a hierarchical topic tree on data cube to define a topic dimension for exploring text information. An example on news data is shown in Figure 6.7.

To improve the TopicCube model, EventCube also generates n-grams to describe the topic. The underneath method is proposed in [41].

6.4 Real-world Cases

6.4.1 NASA: Aviation Safety Report Analysis

One important application of EventCube system is on aviation safety reporting system initiated by NASA. It is a report system used to collect flight report from pilots. Here is the brief summary for the

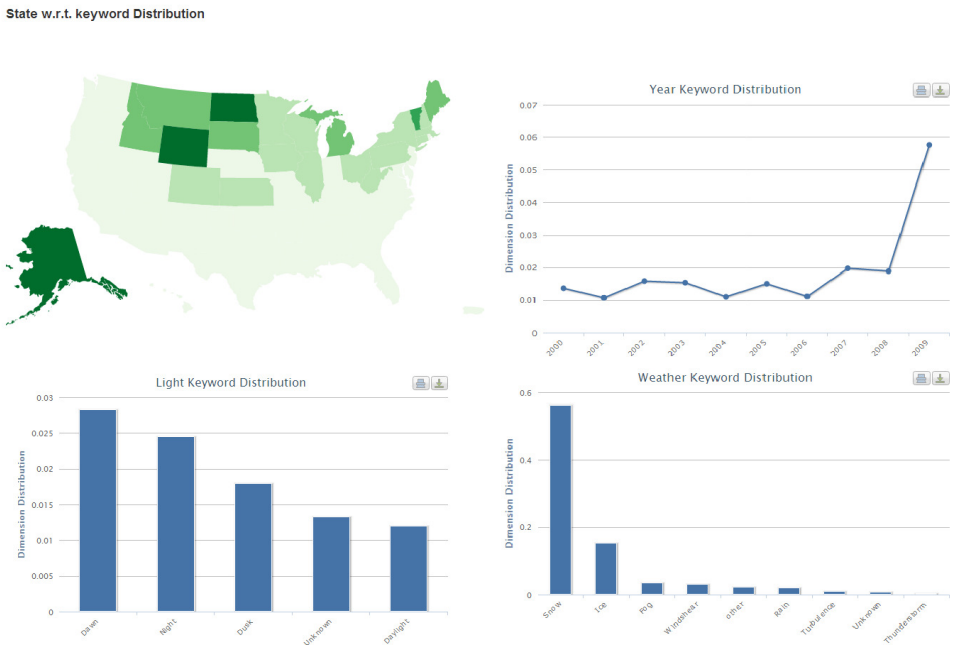


Figure 6.6: Single Dimension Distribution for keyword 'Snow' on the Aviation Safety Reporting Data

dataset.

Dataset: **AVIATION SAFETY REPORT**			
A collection of aviation reports from pilots or maintenance workers from 2007-2011. Each document contains several paragraphs describing the incidents for the flight. More details in https://asrs.arc.nasa.gov/			
# Documents	**# Dimensions**	**Avg. Word Count**	**Text Type**
60K	7	∼200	Reports

Dimension	**# Values**	**Description**
Year	5	Years spanning from 2007 to 2011
State	50	Fifty states in US
Weather	10	Different types of weather condition
Light	5	Different types of light condition
Make Model	430	Different models of the aircraft
Flight Phase	49	Different flight phase, like *taking off*
Event Anomaly	39	Incident type

Table 6.1: Aviation Safety Report

The aforementioned figures are mostly on Aviation Safety Reporting System. We skip the detailed analysis in this section.

6.4.2 Army Research Lab: Counter-Terrorism Report Analysis

Our project is also supported by Army Research Lab via NSCTA (Network Science Collaborative Technology Alliance) program, with many other researchers joined in. Therefore, another interesting dataset we

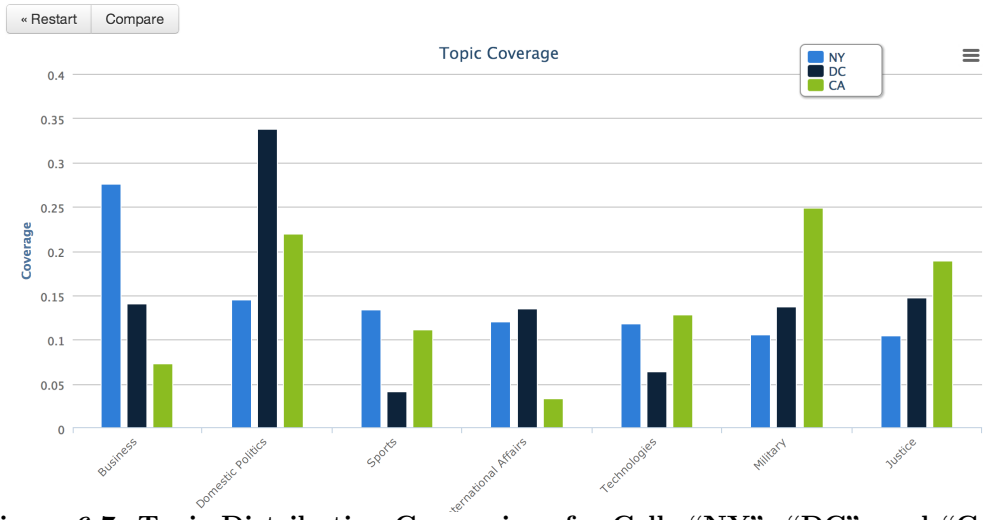


Figure 6.7: Topic Distribution Comparison for Cells "NY", "DC", and "CA"

tested our system on is army counter-terrorism data. Following is the brief summary.

Dataset: **ARMY COUNTER-TERRORISM DATA**			
A collection of short reports from frontier (mid-east) about the target activities. Though the names are anonymized for confidential reasons.			
# Documents	**# Dimensions**	**Avg. Word Count**	**Text Type**
10K	3	∼50	Reports

Dimension	**# Values**	**Description**
Date	200	Different days that the report is received
Location	∼30	Cities that those reports happened
Message Type	5	Types of the received message such as *appear* and *money transfer*

Table 6.2: Army Counter-Terrorism Data

Chapter 7

Conclusion and Future Work

In this thesis, we systematically examine the construction, summarization and mining of text cube. The first line of work we study the problem of text cube construction. Which is, to extract dimension structure from raw text corpus and then assign documents into the multi-dimensional space. The second line of work, we study text consumption. We propose several applications that can conduct effective multi-dimensional text analysis on text cubes, including phrase-based summarization, multi-dimensional topic modeling and structure-rich search.

We present the problem of *dimension-based structure creation* in Chapter 3. We present a semi-supervised bootstrapping framework to extract dimension values using seed sets. With detailed analysis, we show that in a sparse setting, leveraging public web-scale data to understand semantic relations between lexical patterns can effectively alleviate the sparsity. Hence, many entities with very little evidence can be possibly correctly extracted.

After *dimension-based structure creation*, the problem of *cell-based document allocation* is addressed in Chapter 4. We proposed a label-efficient framework that requires only the surface names of different labels, and learns dimension-aware joint embeddings for achieving high accuracies in the allocation process. The experiments validate the effectiveness of the proposed method and its advantages for cell-based document allocation over previous methods.

For cube consumption applications, we first propose CASeOLAP in Chapter 5. We proposes the a solution to support representative phrase mining in multi-dimensional text cubes. It mines top-k representative phrases based on three criteria: integrity, popularity and distinctiveness. An efficient materialization approach is applied to ensure real-time response for large-scale text data.

Lastly, we develop *EventCube*, the comprehensive text cube platform that supports both construction and consumption. This framework provides a tremendous opportunity to conduct multi-dimensional analysis on text and structured data in powerful and flexible ways.

7.1 Potential Future Work

The proposed techniques and the *EventCube* system show promising result towards a powerful text cube platform. However, besides the above discussed research topics in cube construction, summarization and mining, there are still many open challenges yet to be studied. In the last piece of this thesis, we present several future works that extend the scope of text cube to new problems.

7.1.1 Dimension Discovery

This problem is closely related to the cube construction problem discussed in Chapter 3 and 4. In this thesis, we assume domain experts have a set of interested dimensions, *e.g.*, *Location*, *Time* and *Research Domain*, prepared before construction takes place. In many cases, text analysts may not be the domain experts but still interested in obtaining insights from a domain corpus, thus not capable of listing interesting dimensions and seeds associated with them. Even for domain experts, it is common to neglect some interesting aspects of the text and only focus on shallow dimensions. A good example is the *Protest News Analysis* discussed in Section 5.5.4. In this dataset, it is trivial for analysts to design a text cube on *Time*, *Location* and *Incident* dimensions. However, some deeper and more subtle dimensions, such as *Type of Protest*, *Demand of Protest* and *Protester* can be easily missed. Therefore, a data-driven dimension discovery can benefit cube construction and consequently, the cube consumption.

7.1.2 Context-driven Analysis

Context-driven cube analysis stems from the assumption that the ideal summarization and mining result is context-dependent. Here we define *context* as user's browsing history. Browsing history of a user often provides key information of user's attention due to the fact that exploring a multi-dimensional data base can often be a process of narrowing down interests. For example, for the same summarization request of cell $\langle$China, Economy$\rangle$, if the previous request is on cell $\langle$China$\rangle$, it is more reasonable that user is focusing on *Economy* and more economy-related summary is desired. If previous request is on $\langle$Economy$\rangle$, then more China-related summary is more likely desired. The multi-dimensional cube structure provides a sophisticated way to study user's behavior and such behavior-*context* gives opportunities in more intelligent content serving.

7.1.3 Cube-based Content Recommendation

Content recommendation in a large text corpus often refer to the action of recommending documents based on document quality and user interaction history. In text cube, the recommendation system is extended in two ways: 1) rich structural signal can be leveraged to guide better document recommendation. 2cell, as an aggregated content unit, can be recommended as well. For example, while browsing cell $\langle \mathsf{China}, \mathsf{Economy} \rangle$, the most similar or most different sibling cells can be good *cell-level* recommendations for user to further explore. Meanwhile, interesting documents within the same cell or children cells can be good contextual recommendations.

7.1.4 Natural Language Interface

The current *EventCube* system requires user to specify all the dimension values and target function for each request. It can be cumbersome compared to traditional search engines, especially when dimension cardinality is very big. With a natural language interface, analysts can simply type or speak the request, either single-cell requests or comparison requests. Compared to traditional search engine, the predefined structure of text cube can potentially eases the difficulties of understanding user intent. However, the mixture of structural information (*i.e.*, , dimensions and cells) and unstructural information (*i.e.*, keywords) poses unique challenges in understanding the natural language requests.

Bibliography

[1] Aggarwal, C. C., and Zhai, C. A survey of text classification algorithms. *Mining text data* (2012), 163–222.

[2] Agichtein, E., and Gravano, L. Snowball: Extracting relations from large plain-text collections. In *ACM DL* (2000), pp. 85–94.

[3] Baldwin, T., and Kim, S. N. Multiword expressions. *Handbook of Natural Language Processing, second edition. Morgan and Claypool* (2010).

[4] Bedathur, S., Berberich, K., Dittrich, J., Mamoulis, N., and Weikum, G. Interesting-phrase mining for ad-hoc text analytics. *Proceedings of the VLDB Endowment*, 1-2 (2010).

[5] Ben-Yitzhak, O., Golbandi, N., Har'El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E., Sznajder, B., and Yogev, S. Beyond basic faceted search. In *WSDM* (2008).

[6] Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence 35*, 8 (2013), 1798–1828.

[7] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research 3*, Jan (2003), 993–1022.

[8] Brin, S. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*. Springer, 1999, pp. 172–183.

[9] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., and Mitchell, T. M. Toward an architecture for never-ending language learning. In *AAAI* (2010), vol. 5, p. 3.

[10] Carlson, A., Betteridge, J., Wang, R. C., Hruschka Jr, E. R., and Mitchell, T. M. Coupled semi-supervised learning for information extraction. In *WSDM* (2010), pp. 101–110.

[11] Chang, M.-W., Ratinov, L.-A., Roth, D., and Srikumar, V. Importance of semantic representation: Dataless classification. In *AAAI* (2008), vol. 2, pp. 830–835.

[12] Chaudhuri, S., and Dayal, U. An overview of data warehousing and olap technology. *ACM Sigmod record 26*, 1 (1997), 65–74.

[13] Chen, X., Xia, Y., Jin, P., and Carroll, J. Dataless text classification with descriptive lda.

[14] Cohen, W. W., and Singer, Y. A simple, fast, and effective rule learner. In *AAAI* (1999), pp. 335–342.

[15] Danilevsky, M., Wang, C., Desai, N., Ren, X., Guo, J., and Han, J. Automatic construction and ranking of topical keyphrases on collections of short documents. In *SDM* (2014).

[16] Dash, D., Rao, J., Megiddo, N., Ailamaki, A., and Lohman, G. Dynamic faceted search for discovery-driven analysis. In *CIKM* (2008).

[17] Diaz, F., Mitra, B., and Craswell, N. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891* (2016).

[18] Ding, B., Zhao, B., Lin, C. X., Han, J., and Zhai, C. X. Topcells: Keyword-based search of top-k aggregated documents in text cube. In *ICDE* (2010).

[19] Downey, D., Etzioni, O., Soderland, S., and Weld, D. S. Learning text patterns for web information extraction and assessment. In *AAAI* (2004), pp. 50–55.

[20] El-Kishky, A., Song, Y., Wang, C., Voss, C. R., and Han, J. Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment*, 3 (2014).

[21] Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. Web-scale information extraction in knowitall:(preliminary results). In *WWW* (2004), pp. 100–110.

[22] Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. Methods for domain-independent information extraction from the web: An experimental comparison. In *AAAI* (2004), pp. 391–398.

[23] Etzioni, O., Fader, A., Christensen, J., Soderland, S., and Mausam, M. Open information extraction: The second generation. In *IJCAI* (2011), pp. 3–10.

[24] Fader, A., Soderland, S., and Etzioni, O. Identifying relations for open information extraction. In *EMNLP* (2011), pp. 1535–1545.

[25] Freitag, D. Toward general-purpose learning for information extraction. In *ACL* (1998), pp. 404–408.

[26] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery 1*, 1 (1997), 29–53.

[27] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), ACM, pp. 855–864.

[28] Gui, H., Liu, J., Tao, F., Jiang, M., Norick, B., and Han, J. Large-scale embedding learning in heterogeneous event data. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on* (2016), IEEE, pp. 907–912.

[29] Gupta, S., and Manning, C. D. Improved pattern learning for bootstrapped entity extraction. *CoNLL-2014* (2014), 98.

[30] Gupta, S., and Manning, C. D. Spied: Stanford pattern-based information extraction and diagnostics. *Sponsor: Idibon* (2014), 38.

[31] Ha-Thuc, V., and Renders, J.-M. Large-scale hierarchical text classification without labelled data. In *Proceedings of the fourth ACM international conference on Web search and data mining* (2011), ACM, pp. 685–694.

[32] Hearst, M. A. Automatic acquisition of hyponyms from large text corpora. In *Conference on Computational Linguistics* (1992), pp. 539–545.

[33] Hearst, M. A. Clustering versus faceted categories for information exploration. *Communications of the ACM*, 4 (2006).

[34] Inokuchi, A., and Takeda, K. A method for online analytical processing of text data. In *CIKM* (2007).

[35] Ko, Y., and Seo, J. Automatic text categorization by unsupervised learning. In *Proceedings of the 18th conference on Computational linguistics-Volume 1* (2000), Association for Computational Linguistics, pp. 453–459.

[36] Lenci, A. Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics 20*, 1 (2008), 1–31.

[37] Lin, C. X., Ding, B., Han, J., Zhu, F., and Zhao, B. Text cube: Computing ir measures for multidimensional text database analysis. In *ICDM* (2008).

[38] Liu, J., Shang, J., Wang, C., Ren, X., and Han, J. Mining quality phrases from massive text corpora. In *SIGMOD* (2015).

[39] Lu, Y., and Zhai, C. Opinion integration through semi-supervised topic modeling. In *Proceedings of the 17th international conference on World Wide Web* (2008), ACM, pp. 121–130.

[40] Manning, C. D., Raghavan, P., Schütze, H., et al. *Introduction to information retrieval*. Cambridge university press Cambridge, 2008.

[41] Mei, Q., and Zhai, C. A mixture model for contextual text mining. In *Proc. 2006 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'06)* (Philadelphia, PA, Aug. 2006), pp. 649–655.

[42] Mendoza, M., Alegría, E., Maca, M., Cobos, C., and León, E. Multidimensional analysis model for a document warehouse that includes textual measures. *Decision Support Systems* (2015).

[43] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[44] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *EMNLP* (2014), vol. 14, pp. 1532–1543.

[45] Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 701–710.

[46] Poon, H., and Domingos, P. Unsupervised ontology induction from text. In *ACL* (2010), pp. 296–305.

[47] Ravat, F., Teste, O., Tournier, R., and Zurfluh, G. Top_keyword: an aggregation function for textual document olap. In *Data Warehousing and Knowledge Discovery*. Springer, 2008.

[48] Riloff, E. Automatically generating extraction patterns from untagged text. In *AAAI* (1996), pp. 1044–1049.

[49] Riloff, E., Jones, R., et al. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI* (1999), pp. 474–479.

[50] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP* (1995).

[51] Roy, D., Paul, D., Mitra, M., and Garain, U. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608* (2016).

[52] Schmitz, M., Bart, R., Soderland, S., Etzioni, O., et al. Open language learning for information extraction. In *EMNLP-CoNLL* (2012), pp. 523–534.

[53] Sebastiani, F. Machine learning in automated text categorization. *ACM computing surveys (CSUR) 34*, 1 (2002), 1–47.

[54] Simitsis, A., Baid, A., Sismanis, Y., and Reinwald, B. Multidimensional content exploration. *Proceedings of the VLDB Endowment*, 1 (2008).

[55] Song, Y., and Roth, D. On dataless hierarchical text classification. In *AAAI* (2014), vol. 7.

[56] Sutton, C., and McCallum, A. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning* (2006), 93–128.

[57] Tang, J., Qu, M., and Mei, Q. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), ACM, pp. 1165–1174.

[58] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web* (2015), International World Wide Web Conferences Steering Committee, pp. 1067–1077.

[59] Tao, F., Lei, K. H., Han, J., Zhai, C., Cheng, X., Danilevsky, M., Desai, N., Ding, B., Ge, J. G., Ji, H., et al. Eventcube: multi-dimensional search and mining of structured and text data. In *KDD* (2013).

[60] Tao, F., Zhuang, H., Yu, C. W., Wang, Q., Cassidy, T., Kaplan, L. R., Voss, C. R., and Han, J. Multi-dimensional, phrase-based summarization in text cubes. *IEEE Data Eng. Bull. 39*, 3 (2016), 74–84.

[61] Thelen, M., and Riloff, E. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *ACL* (2002), pp. 214–221.

[62] Tunkelang, D. Faceted search. *Synthesis lectures on information concepts, retrieval, and services*, 1 (2009).

[63] Wang, R. C., and Cohen, W. W. Language-independent set expansion of named entities using the web. In *ICDM* (2007), pp. 342–350.

[64] Wang, R. C., and Cohen, W. W. Iterative set expansion of named entities using the web. In *ICDM* (2008), pp. 1091–1096.

[65] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A. J., and Hovy, E. H. Hierarchical attention networks for document classification. In *HLT-NAACL* (2016), pp. 1480–1489.

[66] Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL* (1995), pp. 189–196.

[67] Zhang, C., Tao, F., Gui, H., Shen, J., and Han, J. Taxongen: Constructing topical concept taxonomy by adaptive term embedding and clustering. *Submitted* (2017).

[68] Zhang, D., Zhai, C., and Han, J. Topic cube: Topic modeling for olap on multidimensional text databases. In *SDM* (2009).

[69] Zhang, Z., Iria, J., Brewster, C., and Ciravegna, F. A comparative evaluation of term recognition algorithms. In *LREC* (2008).

[70] Zhao, B., Lin, X., Ding, B., and Han, J. Texplorer: keyword-based object search and exploration in multidimensional text databases. In *CIKM* (2011).